CHALMERS

A Parameter Estimation method for Continuous Time Dynamical Systems based on the Unscented Kalman Filter and Maximum Likelihood

Master's Thesis in Signals and Systems

JOAKIM CARLSSON & CARL NORDHEIM

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011
Master's Thesis 2011:6

Abstract

Acquiring good models of biological and biochemical systems is important in e.g. drug development. These systems are commonly modeled as continuous dynamical systems via ordinary differential equations (ODEs). Measurements on these systems are often taken at discrete time instants, which together with the ODEs yields a system of both continuous and discrete equations. It is usually not realistic to expect the ODEs to provide an exact description of the system. Modeling the system with stochastic differential equations, together with adding noise terms to the measurement equations, is a formal way of including the uncertainties in both the system and the measurement model. This thesis addresses the problem of estimating parameters in this class of models.

The parameter estimation framework that we develop consists of maximum likelihood estimation of the parameters, where the likelihood is approximated via predictions from the unscented Kalman filter. The optimization in the parameter space is performed using a local gradient based method. The gradient is computed analytically by differentiating the filter equations of the unscented Kalman filter with respect to the parameters.

This framework is implemented in Mathematica and validated using two benchmark problems. The performance is compared to that of a corresponding, previously used, framework using the extended Kalman filter. The framework is also compared with frameworks using first and second order finite difference approximations of the gradient. For the two benchmark problems, no improvement is observed by using the unscented Kalman filter instead of the extended Kalman filter. The framework using analytical gradient is computationally faster than those using finite differences for both of the benchmark problems. For one of the benchmark problems, the framework using the analytical gradient gives parameter estimates comparable to both the finite difference frameworks; for the other benchmark problem, the first order difference framework gives considerably worse parameter estimates than the second order difference framework and the framework using the analytical gradient.

Preface

This report completes our master's thesis work on parameter estimation in dynamical systems, the degree project for our Master of Science in Engineering at Chalmers University of Technology. The work has been carried out at Fraunhofer-Chalmers Research Centre for Industrial Mathematics (FCC) in Göteborg, Sweden, in the spring of 2011.

We would like to thank the people who have been involved in this thesis project and helped us through it. First we would like to thank our supervisors at FCC, Jonas Hagmar, Milena Anguelova and head supervisor Mats Jirstrand, for their guidance through the project, and for useful viewpoints on the report.

Furthermore, we would like to thank our supervisor and examiner, Jonas Sjöberg, at the department of Signals and Systems, Chalmers, for his support during this project.

Joakim Carlsson
Carl Nordheim
Göteborg May 23, 2011

Contents

1 Introduction

In Section 1.1, we first give some background to the subject. Then we summarize some of the previous work done by other researchers within the field, and finish by listing the contributions of this thesis. In Section 1.2, we present the outline of the thesis.

1.1 Mathematical models for describing reality and predicting the future

Mathematical models are often used in engineering applications, e.g in control design and for simulation of a process to improve one's understanding of it. Reliable models that describe the actual process well are therefore crucial. Physical processes are often continuous dynamical systems and are naturally modeled with ordinary differential equations (ODEs). Regarding biochemical systems, the actual reactions between individual molecules occur at discrete time instants. But because the number of molecules is large, the variation of concentrations appear continuous, and the system is often modeled by ODEs. An ODE that describes the dynamics of a process can often be derived using physical insight and laws of nature. When doing physical modeling, approximations and simplifications are often introduced to lower the complexity of the model. Examples of such approximations or simplifications can be to assume that a liquid is incompressible, or that the whole mass of a body is contained in its center of mass. Because of the simplifications it is not reasonable to assume that the model will capture the full dynamics of the true process. Even in cases when it is possible to account for all dynamics and thus construct a model that very thoroughly describes the true system, it is often not desired if it results in a very large and complicated model [8]. Instead of using ODEs, it is possible to model the process with a system of stochastic differential equations (SDEs). The stochastic part of the equations can then account for unmodeled dynamics as well as pure stochastic effects that might influence the system. Biological and biochemical processes are often very complex and the exact mechanism behind an observed phenomenon is often not fully understood; thus the process model is uncertain and is naturally modeled by a system of SDEs.

To test whether a model gives a good description of a true system or not, the true system needs to be observed and compared against the model. Observations of a process can be made by measuring certain interesting output signals. These measurements are often taken at discrete time instants [3]. Because the physical or biological process often is inherently time-continuous, a natural description of a process is a system of SDEs describing the dynamics of the process, together with a set of discrete time measurement equations.

When a model of the process has been formulated, the equations contain parameters, e.g. coefficients for reaction rates. Often the parameters are not known and thus have to be estimated. *This master thesis addresses the problem of estimating these parameters.*

For a specific set of parameter values, it is possible to simulate the model and compare the simulation to the measurements obtained from the real process. Naturally we seek values of the parameters that will make the model simulation imitate the real process as well as possible, i.e. predict its future behavior well.

Previous contributions The papers [7], [16] and [17] all study the problem of parameter estimation in a dynamical system using a model structure consisting of a set of non-linear SDEs describing the process dynamics, and measurements taken at discrete time instants with added measurement noise. The general idea in these papers is the same, namely: estimate the system state using some non-linear Kalman filter and calculate the prediction errors and their covariance matrices. Then use the maximum likelihood criterion to perform a prediction error minimization. Thus, the parameter estimation problem is solved as an optimization problem where the negative log-likelihood function is minimized w.r.t. the parameters using gradient based optimization.

The paper [7] uses the continuous-discrete extended Kalman filter (CD-EKF) for state estimation. Both [16] and [17] are to a large extent based on [7], but with some important differences. The paper [16] also uses the CD-EKF as state estimator, but calculates the objective function gradient analytically, whereas [7] approximates it using finite differences. In [17], they use the unscented Kalman filter (UKF) as state estimator and compare it with the EKF. Although they formulate a model where an SDE describes the process dynamics, they discretize the SDE using the Euler method, thus transforming the problem into discrete time. Because they convert the problem into discrete time, i.e. both the process dynamics and measurements are discrete in time, they use the discrete-discrete UKF (DD-UKF) and discrete-discrete EKF (DD-EKF). They claim that the DD-UKF gives better parameter estimates than the DD-EKF. Just like [7], they approximate the objective function gradient with finite differences when performing the optimization.

Because the parameter estimation approach taken by [7], [16] and [17] relies on estimating the system state, the quality of the state estimation will affect the quality of the parameter estimates. It is stated in many papers, e.g. [19] and [4], that the DD-UKF provides better state estimation than the DD-EKF for highly nonlinear systems; [17] claimed that this was probably why the DD-UKF performed better than the DD-EKF in their tests. For the continuous-continuous (CC) case, i.e. the case when both the process dynamics and the measurement model are continuous, [15] states that the CC-UKF is a better state estimator than the CC-EKF in cases where the model nonlinearities and estimation uncertainties are significant.

Contributions of this thesis In this thesis, we use the same model structure and the same parameter estimation framework consisting of maximum likelihood plus state estimator, as [7], [16] and [17] do. The thesis contains two major new contributions that, as far as we know, have not been done before.

- We use the CD-UKF as state estimator. The motivation for using the CD-UKF is that, as discussed above, previous results indicate that the DD-UKF as well as the CC-UKF is a better state estimator than the DD-EKF and the CC-EKF, respectively. Thus it is likely that also the CD-UKF is a better state estimator than the CD-EKF. It is therefore interesting to investigate whether using the CD-UKF as state estimator in the parameter estimation framework can provide better parameter estimates than if the CD-EKF is used.

- We also calculate the gradient of the objective function (log-likelihood function) analytically, which requires differentiation of all the equations in the CD-UKF w.r.t. the parameters. There are two reasons for doing this: investigate whether performing the optimization with the analytical gradient can reduce the computational time of the optimization compared to if a gradient approximation based on finite differences is used. Second, a gradient approximation based on finite differences can be sensitive to numerical errors caused by small errors in the solution of the involved ODEs, as discussed in Section 5.3.

In our benchmark section, we compare the performance of the CD-UKF with analytical gradient against the CD-UKF with numerical gradient and the CD-EKF with analytical gradient on two benchmark problems.

1.2 Outline of thesis

In Section 2 we present the theory that underlies the concepts of the used parameter estimation framework. In Section 3 we give a detailed explanation of the parameter estimation framework and some notes on its implementation. In Section 4 we perform symbolic differentiation of the objective function and the CD-UKF which is needed for using the analytical gradient in the optimization. In Section 5 we test and evaluate the algorithms on two benchmark problems and present the results.

2 Theory behind the parameter estimation framework

This section describes the theoretical background of the parameter estimation framework used in this thesis. Section 2.1 describes the stochastic state space model used to model a system. Section 2.2 describes two different approaches to parameter estimation, gray box models which we use, and white box models that is an alternative. The used measure of goodness of a set of parameters is the likelihood, which is described in Section 2.3. The filtering techniques described in Section 2.4 are necessary to compute the prediction errors and corresponding covariance matrices needed for the likelihood. Finally, in order to find the maximum likelihood estimate of the parameters, optimization has to be performed. This is described in Section 2.5.

2.1 Modeling systems with stochastic differential equations (SDEs)

The continuous-discrete non-linear state space model used is [3, 17, 15]:

$$dx_t = f(x_t, u_t, t, \theta)dt + L(t, \theta)d\beta_t, \qquad t \geq t_0 \tag{2.1}$$

$$y_k = h(x_k, t_k, \theta) + r_k(\theta), \qquad k = 1, 2, ..., N, \tag{2.2}$$

where $x_t \in \mathbb{R}^n$ is the (unobserved) state at time t (x_k is the state at time t_k); $y_k \in \mathbb{R}^m$ are the measurements taken at discrete time instants t_k; $u_t \in \mathbb{R}^r$ is a vector of input signals; $\theta \in \mathbb{R}^p$ is a vector of parameters; $f(\cdot)$ and $h(\cdot)$ are non-linear functions; $L(t, \theta) \in \mathbb{R}^{n \times s}$ is an arbitrary matrix; $\beta_t \in \mathbb{R}^s$ is a standard Wiener process; r_k is Gaussian white noise with covariance matrix $R_k(\theta) \in \mathbb{R}^{m \times m}$; t_k, $k = 0, 1, ..., N$ satisfies $t_0 < t_1 < ... < t_N$. The mean and covariance of the initial state x_0 are $m_0(\theta)$ and $P_0(\theta)$, respectively. The first and second term on the right hand side of (2.1) are denoted the *drift term* and *diffusion term*, respectively.

Equation (2.1) is meaningful only insofar as its integral is defined [3]:

$$x_t - x_{t_0} = \int_{t_0}^{t} f(x_\tau, u_\tau, \tau, \theta)d\tau + \int_{t_0}^{t} L(\tau, \theta)d\beta_\tau.$$

The state space model is continuous-discrete in the sense that the state x_t is continuous in time, whereas the measurements y_k are discrete in time.

We refer to (2.1) as the *process (model)*, (2.2) as the *measurement (model)* and the two together as the *system*.

Markov property For a deterministic process described by an ODE, the evolution of the state is given by

$$\frac{dx_t}{dt} = f(x_t, t).$$

In words, the rate of change of the state x_t at time t depends only on x_t, i.e. the current state and not on earlier values of the state. This means that the current state contains all information needed to predict the future evolution of the state.

For stochastic processes the corresponding property is called *Markov property.* In terms of probability density functions, the Markov property is formulated as

$$p(x_{t_n}|x_{t_1}, x_{t_2}, ..., x_{t_{n-1}}) = p(x_{t_n}|x_{t_{n-1}}).$$

The Markov property states that the conditional probability distribution of the process at time t_n given the whole history of the process up to and including time t_{n-1}, depends only on the process state at time t_{n-1}. Another way of putting this is: the future can be predicted from knowledge of the present.

The stochastic state space model treated in this section is Markov [3].

2.2 System identification and parameter estimation

The method of building mathematical models of real processes from measurement data is called system identification. In this thesis we investigate the problem of estimating unknown parameters in a model structure that has been obtained from physical modeling of the process. To be able to obtain a model with good predictive properties we have to have informative measurement data and a suitable model structure that is able to capture the dynamics of the process.

A process model consisting of a set of ODEs obtained from physical modeling is often called a white-box model. Modeled correctly, white-box models often capture the inherently nonlinear dynamics very well.

The model described in section 2.1, which is sometimes referred to as a gray-box model, is more general than a white-box model. It provides a way of allowing prior physical knowledge to be taken into account and, at the same time, the noise is used for modeling uncertainties in the process dynamics. In many cases the system is not truly stochastic, instead the stochasticity is only used for representing the model uncertainties [14].

Grey-box models allow for a decomposition of the system noise into a process noise term and a measurement noise term. This prediction error decomposition (PED), makes it possible to estimate parameters in gray-box models in a prediction error (PE) setting (described in Section 2.2.2). In white-box models this can only be done in an output error (OE) setting (described below), which tends to give biased and less reproducible results, because random effects are absorbed into the parameter estimates [6].

2.2.1 Parameter estimation in white-box models

When a continuous-discrete non-linear white-box model is postulated, the following dynamic model is used to describe the process:

$$\frac{dx_t}{dt} = f(x_t, u_t, t, \theta), \tag{2.3}$$

$$y_k = h(x_k, t_k, \theta) + r_k, \tag{2.4}$$

$$\mu_k = h(x_k, t_k, \theta). \tag{2.5}$$

This is the same as the state space model presented in Section 2.1, except that the SDE is replaced with an ODE, i.e., there is no process noise. With this model structure one implicitly assumes that there is no uncertainty in the process equations, and hence all deviations from the predicted model output μ_k are assumed to be measurement noise. Therefore, this model structure is called output error (OE).

Parameter estimation in an OE model is typically done using *least squares fitting.* The objective function to be minimized is $V_{ls} = \frac{1}{2}\sum_{k=1}^{N} \epsilon_k(\theta)^T \epsilon_k(\theta)$ where $\epsilon_k(\theta) = y_k - \mu_k(\theta)$ is the output error. The least squares fit is given by the nonlinear optimization problem [11]

$$\underset{\theta\in\Theta}{\text{argmin}}\; V_{ls} = \underset{\theta\in\Theta}{\text{argmin}}\; \frac{1}{2}\sum_{k=1}^{N} \epsilon_k(\theta)^T \epsilon_k(\theta). \tag{2.6}$$

The predicted model outputs ($k = 1, ..., N$) are acquired through $h(\cdot)$ in (2.5), after an initial value simulation of (2.3) over the time period over which the measurements are collected.

The objective function V_{ls}, corresponding to the OE problem formulation, is known for having many local optima. If gradient based local optimization is used, there is a large risk that the minimum found is not the global one, unless the optimization is started very close to the global minimum. In general, if the found parameter estimates give a bad fit, there is no way of knowing if this is due to convergence to a local minimum or an unsatisfactory model structure [13].

One approach to solving these OE problems with many local minima is to use global optimization methods. This approach is taken in [9] and many other studies within the field of systems biology.

2.2.2 Parameter estimation in gray-box models

As was mentioned above, parameter estimation in a gray-box model is approached differently, since this model contains both measurement noise and process noise, thus allowing parameter estimation in a prediction error (PE) setting. In this setting, the differential equations describing the (deterministic) process dynamics are not solved as an initial value simulation. Instead a so called multiple shooting method is used where the differential equations are integrated from one shooting point to the next. A natural choice of shooting points are the time instants of measurements. Information contained in the measurements is then used to reinitialize at proper values at the shooting points.

The multiple shooting point approach often results in a better conditioned problem containing fewer local optima [13]. The method for parameter estimation used in this thesis is a multiple shooting point method and is described in Section 3.

2.3 Maximum likelihood (ML)

The idea of maximum likelihood (ML) estimation [5] is to find the set of model parameters θ that maximizes the probability of generating a given measurement series. More intuitive might be to ask: given a measurement series, what are the most probable (in the mathematical sense) parameters θ? That is however an illegitimate question [11], since the parameters are not statistically distributed, but instead there was in fact one specific set of parameters that generated the data (the measurements on the other hand are drawn from a statistical distribution generated by the model). That is why we instead talk of a likelihood and define the likelihood function for the parameters as the joint probability density of the data given the parameters:

$$L(\theta; \mathscr{Y}_N) = p(\mathscr{Y}_N|\theta), \tag{2.7}$$

where

$$\mathscr{Y}_k = \{y_k, y_{k-1}, ..., y_1\}$$

is a set of measurements. The aim is then to maximize $L(\theta; \mathscr{Y}_N)$ with respect to θ.

Equation (2.7) can be written as a product of conditional probability densities via successive use of the definition of the conditional probability, $P(A \cap B) = P(A|B)P(B)$ [12]:

$$L(\theta; \mathscr{Y}_N) = \left(\prod_{k=2}^{N} p(y_k|\mathscr{Y}_{k-1}, \theta)\right) p(y_1|\theta). \tag{2.8}$$

The state estimators that we use (EKF and UKF) produce Gaussian approximations of the probability densities in (2.8), given by the mean $\mu_k(\theta)$ and covariance $S_k(\theta)$. Using this Gaussian approximation and the *innovation* $\epsilon_k(\theta) = y_k - \mu_k(\theta)$, (2.8) can be written

$$L(\theta; \mathscr{Y}_N) = \prod_{k=1}^{N} \frac{\exp\left(-\frac{1}{2}\epsilon_k(\theta)^T S_k(\theta)^{-1}\epsilon_k(\theta)\right)}{\sqrt{\det(S_k(\theta))}(\sqrt{2\pi})^m},$$

where m is the dimension of y_k as described in Section 2.1. Taking the negative logarithm [5] gives the objective function $V(\theta; \mathscr{Y}_N)$:

$$V(\theta; \mathscr{Y}_N) \equiv -\log(L(\theta; \mathscr{Y}_N)) = \frac{1}{2}\sum_{k=1}^{N} \left(\log(\det(S_k(\theta)) + \epsilon_k(\theta)^T S_k(\theta)^{-1}\epsilon_k(\theta) + m\log(2\pi)\right). \tag{2.9}$$

By minimizing $V(\theta; \mathscr{Y}_N)$ with respect to θ we maximize the likelihood and hence find the ML estimate of the parameters:

$$\hat{\theta} = \underset{\theta\in\Theta}{\operatorname{argmin}}\ V(\theta; \mathscr{Y}_N). \tag{2.10}$$

2.3.1 Relation to least squares

For simplicity, this section uses measurements y_k with only one output signal (so that $y_k^T y_k = y_k^2$). Least squares fitting is an alternative to the maximum likelihood estimation to estimate the parameters θ. It is presented in (2.6), and the following objective function is to be minimized:

$$V_{ls} = \frac{1}{2}\sum_{k=1}^{N} \epsilon_k(\theta)^T \epsilon_k(\theta). \tag{2.11}$$

Assuming some specific properties of the state space model in Equations (2.1) and (2.2), it turns out that minimizing the log-likelihood is equivalent to using the least-squares fit [11]. Suppose that the process noise is zero, i.e. $L(t,\theta) = 0$ in (2.1). Assume further that the measurement noise is constant over time and independent of θ, i.e. $R_k(\theta) = R$ in (2.2). Then it follows that the measurements y_k are independent, so that the joint probability density in (2.7) is a product of the individual probability densities. The likelihood then becomes

$$L(\theta; \mathscr{Y}_N) = \prod_{k=1}^{N} \frac{\exp\left(-\frac{1}{2}\frac{\epsilon_k(\theta)^2}{S}\right)}{\sqrt{2\pi S}},$$

where $S = R$ is the variance of y_k. Taking the negative logarithm gives

$$-\log L(\theta; \mathscr{Y}_N) = \frac{1}{2}\sum_{k=1}^{N} \frac{\epsilon_k(\theta)^2}{S} + \log\sqrt{2\pi S}.$$

Since S is constant, minimizing this is equivalent to minimizing (2.11).

2.3.2 Using multiple measurement series

The likelihood function using one measurement series is defined by $L(\theta; \mathscr{Y}_N) = p(\mathscr{Y}_N|\theta)$ in (2.7). Consider s independent measurement series, $\mathscr{Y}_N^i$, $i = 1, ..., s$, of equal length. The likelihood function using these measurement series is then defined by $L(\theta; \mathscr{Y}_N^1, .., \mathscr{Y}_N^s)$ $= p(\mathscr{Y}_N^1, .., \mathscr{Y}_N^s|\theta) = \prod_{i=1}^s p(\mathscr{Y}_N^i|\theta) = \prod_{i=1}^s L(\theta; \mathscr{Y}_N^i)$. The second equality follows since the measurement series are independent. See [7] for a similar treatment. The objective function $V(\theta; \mathscr{Y}_N)$ now becomes

$$V(\theta; \mathscr{Y}_N^1, .., \mathscr{Y}_N^s) = -\log(\prod_{i=1}^{s} L(\theta; \mathscr{Y}_N^i)) = \sum_{i=1}^{s} -\log L(\theta; \mathscr{Y}_N^i) = \sum_{i=1}^{s} V(\theta; \mathscr{Y}_N^i).$$

2.4 Filtering theory

The problem of estimating the unknown state of a stochastic dynamical system from noisy measurements on the system, given a measurement model and a model of the process dynamics, is called the filtering problem [3].

In this thesis we are mainly interested in estimating the state of the the continuous-discrete state space model given in Section 2.1. The recursive real-time estimation algorithms for these filtering problems are called (optimal) continuous-discrete filters.

The formal solution to the problem of optimal non-linear filtering is well known, and consists of sequential solving of the Kolmogorov forward equation and the application of Bayes' rule. To solve the Kolmogorov partial differential equation is however in the general case impossible, because it would require an infinite amount of computations. In some special cases the analytical closed form solution exists. This is for example the case when the system dynamics are linear, and the measurement and process noise are Gaussian. The continuous-discrete Kalman filter is then the optimal filter [14].

If, on the other hand, the dynamics are nonlinear and/or the noise is non-Gaussian, one has to resort to approximate solutions. The extended Kalman filter (EKF) linearizes the non-linearities and then uses the Kalman filter. The unscented Kalman filter (UKF) approximates the filtering distributions by Gaussian distributions, i.e. Gaussian noise and Gaussian state and output distributions.

2.4.1 Continuous-discrete Kalman filter (CD-KF)

As mentioned above, the Kalman filter is the optimal filter when the system dynamics are linear and the measurement and process noise are Gaussian. It is optimal in the sense that it gives the minimum variance estimate of the system state [3]. The purpose of optimal filtering is to compute the conditional distribution of the system state $x(t)$, given the history of the measurements up to time t:

$$p(x_t|\mathscr{Y}_k, \theta), \qquad t_k < t < t_{k+1}.$$

Because the noise is Gaussian and the system dynamics are linear, the filtering distributions will be Gaussian. The Gaussian density is completely characterized by its first and second moments, or equivalently, its mean vector m_t and covariance matrix P_t.

The continuous-discrete Kalman filter consists of the following initialization, prediction and update steps:

- *At the time of initialization, initial values for the mean $m_0(\theta)$ and the state covariance $P_0(\theta)$ have to be provided.*
- *At the prediction step, the differential equations*

$$\frac{dm_t}{dt} = F(t,\theta)m_t + B(t,\theta)u_t \tag{2.12}$$

$$\frac{dP_t}{dt} = F(t,\theta)P_t + P_tF(t,\theta)^T + L(t,\theta)L(t,\theta)^T \tag{2.13}$$

  are integrated from time t_{k-1}, with initial conditions $m_{t_{k-1}} = m_{k-1}$, $P_{t_{k-1}} = P_{k-1}$, to time instant t_k. The predicted mean and covariance are $m_k^- = m_{t_k}$and $P_k^- = P_{t_k}$, respectively. $F(t,\theta) \in \mathbb{R}^{n\times n}$ is the process matrix and $B(t,\theta) \in \mathbb{R}^{n\times r}$ is the input matrix. $F(\cdot)$ and $B(\cdot)$ replaces $f(\cdot)$ in Equation (2.1) *for a linear state space model.*

- *At the update step, the measurement and its covariance are taken into account to improve the state estimates:*

$$\mu_k = H_k(\theta)m_k^- \tag{2.14}$$

$$S_k = H_k(\theta)P_k^- H_k(\theta)^T + R_k(\theta) \tag{2.15}$$

$$K_k = P_k^- H_k(\theta)^T S_k^{-1} \tag{2.16}$$

$$m_k = m_k^- + K_k(y_k - \mu_k) \tag{2.17}$$

$$P_k = P_k^- - K_k S_k K_k^T. \tag{2.18}$$

S_k is the covariance matrix of measurement k; K_k is the Kalman filter gain matrix; m_k is the updated state mean at time t_k and P_k is the updated state mean at time t_k. $H_k(\theta) \in \mathbb{R}^{m\times n}$ is the measurement matrix, that replaces $h(x_k, t_k, \theta)$ in Equation (2.2) *for a linear state space model.*

The Kalman filter is a so called recursive filter, and thus, the amount of computations is the same for all time steps, i.e. previously processed data need not be stored and processed again. Instead, only the most recent data is used in the calculations. This is a consequence of the state space model having Markov properties.

2.4.2 Prediction and update step of Kalman filters

The continuous-discrete Kalman filter in Section 2.4.1 is a *state estimator*. The continuous-discrete extended Kalman filter (CD-EKF) and the continuous discrete unscented Kalman filter (CD-UKF), described in Section 2.4.3 and 2.4.5, respectively, are also state estimators. The computations of the CD-KF, CD-EKF and CD-UKF are divided into a prediction step and an update step. Here we give an interpretation of this formalism, and describe in words what the Kalman filter does:

- In the prediction step a prediction of the system state is formed by propagating the state through the deterministic part of the SDE, from the time instant of one measurement to the time instant of the next.
- The update step is performed at the time instants of the measurements. The state estimation is improved using the additional information contained in the measurement. This update correction gives a jump in the state estimate, leading to a state estimation trajectory that is piecewise continuous (see Figure 1).

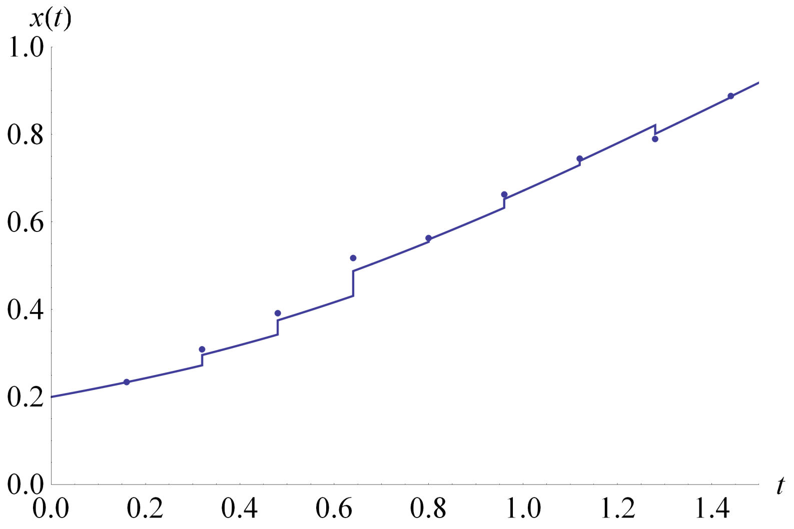


Figure 1: Piecewise continuous state trajectory where jumps at the instants of measurements can be seen. Note that the directions of the jumps are always towards the measurements.

When using a Kalman filter (KF, EKF or UKF), both the measurement and model prediction are used to form the state estimate. Thus a better estimate is achieved than if just one of them is used. The relative magnitude between process and measurement noise will affect the filter's amount of correction at the update steps. If the process noise is large compared to the measurement noise, measurements are greatly trusted, and a large correction is made. If at the other hand the process noise is small compared to the measurement noise the model is trusted more than the measurements and a smaller correction is made. The amount of correction is controlled by the Kalman gain K_k, Equation (2.16).

2.4.3 Continuous-discrete extended Kalman filter (CD-EKF)

The most frequently used method for state estimation when the drift function f, or the measurement function h, is nonlinear, is to use the extended Kalman filter (EKF). It uses a Taylor series approximation of the non-linear functions and forms a Gaussian process approximation [14].

The first order CD-EKF uses a first order Taylor expansion, i.e. linearization, of f and h to get their Jacobian matrices. It consists of the following initialization, prediction and update steps:

- *At the time of initialization, initial values for the mean $m_0(\theta)$ and the state covariance $P_0(\theta)$ have to be provided.*
- *At the prediction step, the differential equations*

$$\frac{dm_t}{dt} = f(m_t, u_t, t, \theta) \tag{2.19}$$

$$\frac{dP_t}{dt} = F_x(m_t, u_t, t, \theta)P_t + P_t F_x(m_t, u_t, t, \theta)^T + L(t, \theta)L(t, \theta)^T \tag{2.20}$$

are integrated from time t_{k-1}, with initial conditions $m_{t_{k-1}} = m_{k-1}$, $P_{t_{k-1}} = P_{k-1}$, to time instant t_k. The predicted mean and covariance are given as $m_k^- = m_{t_k}$ and $P_k^- = P_{t_k}$, respectively. $F_x(m_t, u_t, t, \theta) \in \mathbb{R}^{n\times n}$ is the Jacobian matrix of f with elements $[F_x(m_t, u_t, t, \theta)]_{i,j} = \frac{\partial f_i(x,u_t,t,\theta)}{\partial x_j}|_{x=m_t}$.

- *At the update step, the measurement and its covariance are taken into account to improve the state estimates:*

$$\mu_k = h(m_k^-, t_k, \theta) \tag{2.21}$$

$$S_k = H_x(m_k^-, t_k, \theta)P_k^- H_x(m_k^-, t_k, \theta)^T + R_k(\theta) \tag{2.22}$$

$$K_k = P_k^- H_x(m_k^-, t_k, \theta)^T S_k^{-1} \tag{2.23}$$

$$m_k = m_k^- + K_k (y_k - \mu_k) \tag{2.24}$$

$$P_k = P_k^- - K_k S_k K_k^T. \tag{2.25}$$

S_k is the covariance matrix of measurement k; K_k is the Kalman filter gain matrix; m_k is the updated state mean at time t_k and P_k is the updated state mean at time t_k. $H_x(m_k^-, t_k, \theta) \in \mathbb{R}^{m\times n}$ is the Jacobian matrix of h with elements $[H_x(m_k^-, t_k, \theta)]_{i,j} = \frac{\partial h_i(x,t_k,\theta)}{\partial x_j}|_{x=m_k^-}$.

In (2.19) and (2.24) f and h are used directly in the filter. Apart from this, the EKF uses the ordinary (linear) Kalman filter equations, but with $F(\cdot)$ and $H_k(\cdot)$ replaced with the Jacobians of $f(\cdot)$ and $h(\cdot)$, respectively.

2.4.4 Unscented Transform (UT)

The unscented transform (UT) [15, 19] is used in the CD-UKF. It is a method that can be used for forming a Gaussian approximation of the joint distribution of random variables x and y, when the random variable y is obtained by a nonlinear transformation of the Gaussian random variable x as follows:

$$x \sim N(m, P)$$
$$y = g(x),$$

where $x \in \mathbb{R}^n; y \in \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ is a non-linear function.

The idea of the UT is to form a set of so called sigma points, that capture the mean and covariance of x exactly. These are propagated through the nonlinear function g, and are then used to approximate the mean and covariance of the transformed variable y. The Gaussian approximation of the joint distribution of x and y is now

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim N\left(\begin{bmatrix} m \\ \mu_U \end{bmatrix}, \begin{bmatrix} P & C_U \\ C_U^T & S_U \end{bmatrix}\right), \tag{2.26}$$

where m and P are the mean vector and covariance matrix of x; μ_U and S_U are the approximate mean vector and covariance matrix of y and C_U is the approximate cross-covariance of x and y.

Below the UT is presented in two different forms, denoted *Summation form* and *Matrix form*. The latter is a more compact notation where the weighted sums over sigma points are replaced by equivalent matrix expressions.

Summation form The UT in summation form is defined by the following four steps:

1. Compute the $d \equiv 2n+1$ sigma points

$$x^{(i)} = \begin{cases} m & \text{if } i = 0, \\ m + \sqrt{n+\lambda} A_i & \text{if } i = 1, ..., n, \\ m - \sqrt{n+\lambda} A_{i-n} & \text{if } i = n+1, ..., 2n, \end{cases} \tag{2.27}$$

and the corresponding weights

$$W_0^{(m)} = \frac{\lambda}{n+\lambda},$$

$$\begin{aligned} W_0^{(c)} &= \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta, \\ W_i^{(m)} &= \frac{1}{2(n+\lambda)}, \qquad i = 1, ..., 2n, \\ W_i^{(c)} &= \frac{1}{2(n+\lambda)}, \qquad i = 1, ..., 2n. \end{aligned}$$

The matrix A is the lower triangular matrix in the Cholesky decomposition of P, i.e.

$$AA^T = P \tag{2.28}$$

with A lower triangular. A_i denotes the ith column of A. The Cholesky decomposition is only defined for positive definite matrices P.

The constants α, β and κ are used as parameters of the method. α determines the spread of the sigma points around m and is usually set to $10^{-4} \leq \alpha \leq 1$. κ is a secondary scaling parameter which is usually set to 0. β is used to incorporate prior knowledge of the distribution of x (for Gaussian distributions, $\beta = 2$ is optimal) [18].

The scalar λ is defined by $\lambda = \alpha^2(n+\kappa) - n$.

Note that the sigma points in (2.27) are sampled symmetrically around the mean m.

2. Transform the sigma points according to

$$y^{(i)} = g(x^{(i)}), \qquad i = 0, ..., 2n.$$

3. Compute mean and covariance estimates for y as

$$\mu_U = \sum_{i=0}^{2n} W_i^{(m)} y^{(i)},$$

$$S_U = \sum_{i=0}^{2n} W_i^{(c)} (y^{(i)} - \mu_U)(y^{(i)} - \mu_U)^T.$$

4. Estimate the cross-covariance of x and y as

$$C_U = \sum_{i=0}^{2n} W_i^{(c)} (x^{(i)} - m)(y^{(i)} - \mu_U)^T.$$

Matrix Form In the matrix form of the UT, the matrix X and the function $g_M(X)$ are used. They are defined as follows:

- The matrix $X \in \mathbb{R}^{n \times d}$ is defined by

$$X_i = x^{(i)}, \qquad i = 0, .., 2n, \tag{2.29}$$

where X_i denotes the ith column of X and $x^{(i)}$ is the ith sigma point.

- The function $g_M : \mathbb{R}^{n \times d} \to \mathbb{R}^{m \times d}$ is defined by

$$(g_M(X))_i = g(X_i), \qquad i = 0, .., 2n, \tag{2.30}$$

where $(g_M(X))_i$ denotes the ith column of $g_M(X)$.

The UT in matrix form is now written

$$X = \left[\begin{array}{ccc} m & ... & m \end{array} \right] + \sqrt{c} \left[\begin{array}{ccc} 0 & A & -A \end{array} \right] \tag{2.31}$$

$$Y = g_M(X) \tag{2.32}$$

$$\mu_U = Y w_m \tag{2.33}$$

$$S_U = Y W Y^T \tag{2.34}$$

$$C_U = X W Y^T, \tag{2.35}$$

where $c = \alpha^2(n + \kappa)$; the vector $w_m \in \mathbb{R}^n$ of sigma point weights is defined by

$$w_m = \left[\begin{array}{ccc} W_m^{(0)} & ... & W_m^{(2n)} \end{array} \right]^T,$$

and the matrix W is defined by

$$W = \left(I - \left[\begin{array}{ccc} w_m & ... & w_m \end{array} \right] \right) \times \mathrm{diag}\left(\left[\begin{array}{ccc} W_c^{(0)} & ... & W_c^{(2n)} \end{array} \right] \right) \times \left(I - \left[\begin{array}{ccc} w_m & ... & w_m \end{array} \right] \right)^T,$$

where $\mathrm{diag}(v)$ is a diagonal matrix with v on the diagonal.

Example 1. *In this example we perform a nonlinear transformation of a Gaussian random variable and approximates the distribution of the transformed random variable using the UT, which is used in the UKF, and the linear approximation used in the EKF. We compare the results with the true distribution of the transformed random variable.*

Consider the random vector $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ *distributed as follows:*

$$x \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \pi^2/4 & 0 \\ 0 & 1 \end{bmatrix} \right), \tag{2.36}$$

The distribution of x is illustrated in Figure 2(a). x is transformed to $y = g(x)$ *according to*

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -sin(x_1) \end{bmatrix}. \tag{2.37}$$

Using the UT a set of 5 (2n + 1) sigma points are formed to capture the mean and covariance of x according to Equation (2.31), as indicated in the figure. The sigma points are transformed according to (2.32). Estimates of the mean and covariance of y are calculated according to (2.33) and (2.34), respectively.

For the linear approximation, the estimate of the mean of y is $\mu_{lin} = g(x)$*, analogous to (2.21), and the estimate of the covariance of y is* $S_{lin} = G_x(x)PG_x(x)^T$ *where* $G_x(x)$ *is the Jacobian of* $g(x)$*. This is analogous to (2.22), but without the noise term.*

In this example, both the UT and the linear approximation calculate the mean of y correctly. The covariance approximations of y are the follwing:

- *For the UT:*
$$S_{UT} = \begin{bmatrix} 1 & 0 \\ 0 & 0.3166 \end{bmatrix}. \tag{2.38}$$

- *For the linear approximation:*
$$S_{lin} = \begin{bmatrix} 1 & 0 \\ 0 & 2.467 \end{bmatrix}. \tag{2.39}$$

- *And the true covariance of y is*
$$S_y = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} - \frac{1}{2}e^{-\pi^2/2} \end{bmatrix} \approx \begin{bmatrix} 1 & 0 \\ 0 & 0.4964 \end{bmatrix}. \tag{2.40}$$

As can be seen from the covariance matrices, the UT underestimates the variance of y_2 *slightly, and the linear approximation overestimates it quite a lot. This is also illustrated in Figure 2(b), where the true distribution of the transformed random variable y is shown together with the approximated distributions given by the UT and the linear approximation.*

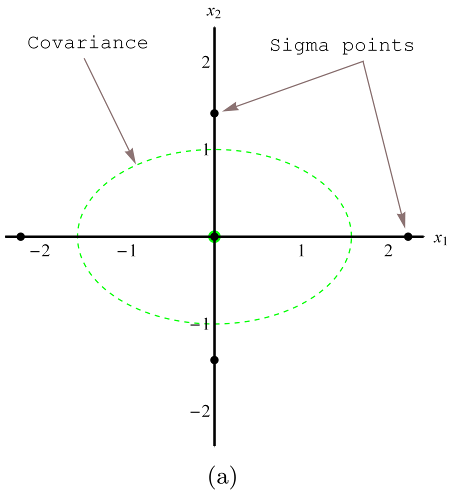
A few remarks are in place to explain some of the details concerning Figure 2:

- *To calculate the true mean and covariance of y, we make use of the formula [12]*

$$E(g(x_1, x_2)) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x_1', x_2') f_x(x_1', x_2') dx_1' dx_2', \qquad (2.41)$$

  where $E(\cdot)$ is the expected value and $f_x(\cdot)$ is the pdf of x. Using (2.41), it is possible to compute all the moments necessary to find the mean vector and covariance matrix of y.

- *The covariance ellipses in Figure 2 are constructed from the respective covariance matrix and mean vector. The directions of the semi-axes are given by the eigenvectors to the covariance matrix, and the length of the semi-axes are given by the square root of the corresponding eigenvalues. The ellipses are centered at the mean vector. Constructed in this way, the ellipses are contour lines to the probability density function (pdf), when the distribution is bivariate normal [1]. This holds for x, the linear approximation of y and the UT approximation of y, but y on the other hand is obtained from a non-linear transformation of a Gaussian variable (x), and is thus not Gaussian. The covariance ellipse will in this case not be a contour line to the pdf, but is still used to illustrate the distribution.*



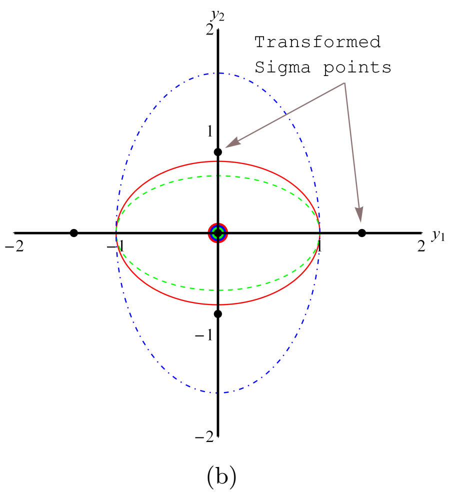


Figure 2: Illustration of distributions and sigma points from Example 1. a) Distribution of x. The mean of the distribution is indicated in the center of the figure, but coincides with the zeroth sigma point and is not clearly visible. The sigma points are indicated with dots and are symmetrically sampled around the mean. b) Distribution of y and the linear approximation and the UT approximation of the distribution of y. The dashed covariance (smallest) ellipse belongs to the UT approximation, the dashed-dotted (largest) covariance ellipse belongs to the linear approximation and the full line covariance ellipse belongs to y. The mean of y, the mean of the UT approximation and linear approximation of y, and one of the sigma points coincide at the origin. The transformed sigma points happen to be symmetrical around the mean, but in general the sigma points are not symmetrical around the mean after a non-linear transformation.

2.4.5 Continuous-discrete Unscented Kalman Filter (CD-UKF)

The original UKF is concerned with the discrete-discrete state space model where, in contrast to the state space model described in Section 2.1, the state is discrete in time [4]. The continuous-discrete UKF presented here is derived from the original UKF in [15].

The continuous-discrete UKF uses the UT to propagate random variables through $f(\cdot)$ and $h(\cdot)$. This is particularly clear in Equations (2.44)-(2.48). The sigma matrix X_k^- is formed, propagated through $h_M(\cdot)$, and weighted to acquire μ_k, S_k and C_k. Compare with (2.31)-(2.35), but note that the term R_k in (2.47) does not originate from the propagation through $h_M(\cdot)$, and is thus not a result of the UT; rather, it is added to account for the measurement noise in (2.2). It is also quite clear how the UT is used in (2.42), where the sigma matrix X is formed, transformed via $f_M(\cdot)$ and weighted with w_m (compare with Equations (2.31)-(2.33)).

- *Prediction step: Integrate*

$$\frac{dm_t}{dt} = f_M(X_t, u_t, t, \theta)w_m \tag{2.42}$$

$$\frac{dP_t}{dt} = X_t W f_M(X_t, u_t, t, \theta)^T + f_M(X_t, u_t, t, \theta) W X_t^T + L(t, \theta)L(t, \theta)^T \tag{2.43}$$

from time t_{k-1} to t_k. X_t is the sigma point matrix X in (2.31) at time t and $f_M(\cdot)$ is a mapping of $f(\cdot)$ on the sigma points, analogous to $g_M(\cdot)$ in (2.30). The initial conditions are $m_{t_{k-1}} = m_{k-1}$ and $P_{t_{k-1}} = P_{k-1}$. The predicted state mean m_k^- and state covariance P_k^- are given as m_{t_k} and P_{t_k}.

- *Update step: Compute the predicted matrix of sigma points X_k^-; the transformed sigma matrix Y_k^-; the mean μ_k and covariance S_k of the measurement; the cross-covariance of the state and measurement C_k; the Kalman gain K_k, and the updated mean m_k and covariance P_k of the state:*

$$X_k^- = \begin{pmatrix} m_k^- & \dots & m_k^- \end{pmatrix} + \sqrt{c}\begin{pmatrix} 0 & A_k^- & -A_k^- \end{pmatrix} \tag{2.44}$$

$$Y_k^- = h_M(X_k^-, t_k, \theta), \tag{2.45}$$

$$\mu_k = Y_k^- w_m \tag{2.46}$$

$$S_k = Y_k^- W (Y_k^-)^T + R_k(\theta) \tag{2.47}$$

$$C_k = X_k^- W (Y_k^-)^T \tag{2.48}$$

$$K_k = C_k S_k^{-1} \tag{2.49}$$

$$m_k = m_k^- + K_k(y_k - \mu_k) \tag{2.50}$$

$$P_k = P_k^- - K_k S_k K_k^T, \tag{2.51}$$

where A_k^- is the lower triangular Cholesky factor of P_k^-, analogous to the relation between A and P in Section 2.4.4 and $h_M(\cdot)$ is a mapping of $h(\cdot)$ on the sigma points, analogous to $f_M(\cdot)$.

2.5 Optimization

The estimates of the parameters θ are obtained by solving the unconstrained minimization problem (2.10), which is restated here:

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \{V(\theta; \mathscr{Y}_N)\},$$

where $\hat{\theta}$ is the estimate of the parameters. Given the objective function $V(\theta; \mathscr{Y}_N)$, this corresponds to minimization in the multidimensional parameter space, with one dimension for each component of θ. There are many algorithms available for this kind of problem, for example Newton's method and various quasi-Newton methods, which are all local methods. In this thesis, we have used the BFGS method, which is a quasi-Newton method. With Newton's method and the BFGS method, a starting guess is supplied, and then the algorithm iteratively takes steps in the parameter space to get closer to the minimum. Using Newton's method, the gradient and the Hessian of the objective function are calculated either symbolically or by finite differences in order to find the next step. In the BFGS method, the Hessian is not calculated symbolically or by finite differences, but instead approximated using information from function and gradient evaluations from previous steps. In both Newton's method and the BFGS method, the search direction is computed from the gradient and Hessian approximations. In addition to that, a line search can be performed to determine how far to go in that direction. In this thesis, we use the BFGS method together with a line search. See [20] for more details of how these methods are implemented in Mathematica.

2.5.1 Finite difference approximations of derivatives

When using finite differences to approximate the derivative (gradient), one can use different types of approximations, first order, second order and so on. A kth order approximation means that the truncation error is $\mathcal{O}(h^k)$ where h is the step size.

The first order approximation of the derivative of a single variable function is:

$$\left.\frac{df(x)}{dx}\right|_{x=a} = \frac{f(a+h)-f(a)}{h}. \tag{2.52}$$

The extension to partial derivative w.r.t. one variable for a multivariable function is:

$$\left.\frac{\partial f(x)}{\partial x_l}\right|_{(x_1,...,x_l,...,x_p)=(a_1,...,a_l,...,a_p)} = \frac{f(a_1,...,a_l+h,...,a_p)-f(a_1,...,a_l,...,a_p)}{h}. \tag{2.53}$$

The gradient is composed of partial derivatives with respect to each of the p parameters. It can be inferred from the above equation that the gradient evaluation requires $p+1$ distinct function evaluations. For sufficiently large values of p, the number of function evaluations increases approximately linearly with p.

The second order approximation of the derivative of a single variable function is:

$$\left.\frac{df(x)}{dx}\right|_{x=a} = \frac{f(a+h)-f(a-h)}{2h}. \tag{2.54}$$

And the extension to partial derivative w.r.t. one variable for a multi variable function is:

$$\left.\frac{\partial f(x)}{\partial x_l}\right|_{(x_1,...,x_l,...,x_p)=(a_1,...,a_l,...,a_p)} = \frac{f(a_1,...,a_l+h,...,a_p)-f(a_1,...,a_l-h,...,a_p)}{2h}. \tag{2.55}$$

It can be inferred from the above equation that the gradient in this case requires $2p$ distinct function evaluations. The number of evaluations thus increases linearly with the number of parameters.

Letting the step size h go to zero in the above equations, the approximations will approach the true derivatives, by definition of the derivative.

3 Method for parameter estimation

In this section we present the method used in this thesis to estimate parameters in a stochastic state space model (see Section 2.1) also known as a stochastic gray-box model. The method used is essentially the same as the one treated in [7]. In Section 1 we point out the differences between our approach and theirs.

3.1 Using Maximum likelihood together with a state estimator to estimate the model parameters θ

To estimate the parameters θ we perform a prediction error minimization by using the maximum likelihood (ML) method (see Section 2.3). Maximizing the likelihood function is equivalent to finding the set of parameters that is most likely to have produced the given measurement series [5]. As was stated in section 2.3, maximizing the likelihood function is equivalent to minimizing the negative log-likelihood function. The negative log-likelihood is our choice of objective function and is restated here:

$$V(\theta; \mathscr{Y}_N)) \equiv -\ln(L(\theta; \mathscr{Y}_N)) = \frac{1}{2}\sum_{k=1}^{N}\left(\ln(\det(S_k) + \epsilon_k^T S_k^{-1}\epsilon_k + m \cdot \ln(2\pi)\right). \tag{3.1}$$

By minimizing $V(\theta; \mathscr{Y}_N)$ with respect to θ we maximize the likelihood and hence find the ML estimates of the parameters:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}}\{V(\theta; \mathscr{Y}_N)\}. \tag{3.2}$$

Thus, the parameter estimation problem is solved as an optimization problem parametrized by the parameters θ.

The general idea of our method can be summarized as:

- For a given parameter vector θ, Kalman filtering is used to estimate the system state and will for each measurement k, obtained at time t_k, provide the innovation ϵ_k and its covariance matrix S_k, needed to calculate the objective function $V(\theta; \mathscr{Y}_N)$ in (3.1). The system state will then be a function of the current parameter vector and the given measurement sequence

$$\mathscr{Y}_N = [y_1, y_2, ..., y_{N-1}, y_N]. \tag{3.3}$$

  To get the innovation ϵ_k and its covariance matrix S_k for each time t_k, a filter has to be run recursively. The Kalman filter is the choice if the dynamics is linear, and EKF or UKF is used if it is nonlinear. For details and filter equations, see Section 2.4

- When the filter has been run from start time t_0 to final time t_N, the objective function value $V(\theta; \mathscr{Y}_N)$ can be calculated for the particular vector θ. Now the optimization problem stated in (3.2) is solved as follows: search for the parameter vector that minimizes $V(\theta; \mathscr{Y}_N)$, by using gradient based optimization w.r.t. the parameter vector θ.

As mentioned above, several different filters can be used. In this thesis the focus is on the CD-UKF, described in Section 2.4.5.

Also, as stated above we use gradient-based optimization to minimize $V(\theta; \mathscr{Y}_N)$, thus the objective function gradient is needed in the optimization. The most straightforward approach is to approximate the gradient with finite differences. We have, however, calculated it analytically for the CD-UKF. This was done by differentiating $V(\theta; \mathscr{Y}_N)$ and the whole CD-UKF scheme, i.e. all equations with respect to θ. This is done in Section 4. In the benchmark section (Section 5) we compare the performance of the analytic gradient with the finite difference approximated gradient.

When both the CD-UKF filter and the differentiated filter are available (or some other filter for that matter), the optimization procedure is the following:

- Choose an initial guess of the parameter vector θ. Often prior knowledge of the system or simple experiments involving it can provide a good initial guess.
- For this particular parameter vector $V(\theta; \mathscr{Y}_N)$ is computed as described above. A good θ will give a small function value. If the analytical gradient is used, the gradient of the objective function (see Section 4.1) is computed by running the differentiated scheme in the same way as the filter scheme.
- The computed gradient is used in the optimization to find new and better parameter estimates. The optimization method used is of quasi-Newton type and is described in Section 2.5.
- This search procedure is iterated until a termination criterion is fulfilled, and thus a local minimum has been reached. The corresponding minimum point, $\hat{\theta}$, is then the proposed parameter vector.

3.2 Implementation

The programming language used in this thesis is *Mathematica*. A clear benefit of using Mathematica is that it is possible to do calculations symbolically, rather than just numerically. Thus for example when differentiating f with respect to θ, we can do so analytically with the particular f used. In Section 3.2.1 we present the structure, in terms of functions used, for the parameter estimation framework. Section 3.2.2 covers some details on how the optimization is performed in Mathematica.

3.2.1 The implemented framework for parameter estimation

The parameter estimation framework makes use of two function packages, described below, which are not part of the standard Mathematica language.

DynamicSystems This is a function package developed at Fraunhofer-Chalmers Research Centre for Industrial Mathematics (FCC). We make use of three functions in this package:

- *DynamicModelObject* creates a DynamicModel instance. A DynamicModel instance contains the particular system used, i.e. process equations (Equation (2.1)) and measurement equations (Equation (2.2)).
- *DynamicModelObserve* creates simulated measurements from a DynamicModel instance at specified time instants.
- *DynamicModelSolvePredictor* runs a filter and returns the value of the objective function (see Equation (3.1)) in the case the filter is CD-EKF or CD-UKF. If the filter that is run is the differentiated CD-EKF or the differentiated CD-UKF, DynamicModelSolvePredictor returns the gradient of the objective function. Possible options include different values for the initial state covariance P_0, and for the extra method parameters of the CD-UKF: α, β and κ. DynamicModelSolvePredictor uses the x_0 specified in the DynamicModel instance as m_0.

KalmanFilters This is a function package that we have developed. It contains functions that are used by DynamicModelSolvePredictor described above. When DynamicModelSolvePredictor runs a filter, it does so in two steps. First it calls a function that defines functions such as $f(\cdot)$ and $h(\cdot)$ (see Equations (2.1)-(2.2)). This is referred to as the preparation step. Then it calls a function that runs the actual filter, i.e. performs the loop of predictions and updates, and returns the value of the objective function (see Equation (3.1)) or the value of the gradient of the objective function.

The functions that perform the preparation step are *prepareCDEKF*, *prepareCDUKF*, *prepareCDEKFD* and *prepareCDUKFD*. prepareCDEKF prepares the CD-EKF filter; prepareCDEKFD prepares the differentiated CD-EKF filter, and analogously for the CD-UKF. The corresponding functions for performing the loop of predictions and updates are *loopCDEKF*, *loopCDUKF*, *loopCDEKFD* and *loopCDUKFD*.

3.2.2 Minimization in Mathematica

The built-in function *FindMinimum* is used to perform local minimization. Its arguments are the function to be minimized as well as a starting guess. It can be supplied with various options specifying which methods to use etc. As described in Section 2.5, we use the BFGS method together with a line search, which is also the default setting for FindMinimum. The BFGS method uses the gradient of the objective function in determining the search direction. The gradient can be computed by FindMinimum using finite differences. It can also be supplied to FindMinimum in the form of a function. The latter approach is used in this thesis since we compute the gradient analytically.

Any iterative optimization algorithm will need a termination criterion, determining when the solution is accurate enough to stop the optimization. FindMinimum includes two options for determining this: *PrecisionGoal* and *AccuracyGoal*. By specifying the values p and a of the PrecisionGoal and AccuracyGoal options respectively, FindMinimum attempts to use the following termination criterion:

$$||\theta_k - \theta^*|| \leq \max(10^{-a}, 10^{-p}||\theta_k||) \text{ and } ||\nabla f(\theta_k)|| \leq 10^{-a},$$

where θ_k is the value of θ in the k:th step and θ^* is the minimum point. Of course, θ^* is not known, and the quantity $||\theta_k - \theta^*||$ has to be estimated. To our experience, FindMinimum does not always issue a warning if it does not succeed in reaching the criterion $||\nabla f(\theta_k)|| \leq 10^{-a}$.

If FindMinimum cannot reach the termination conditions, it is our experience that it will eventually stop and give a message that the line search cannot find a sufficient decrease in the function. To choose a very strict termination condition, for example $a = \infty$, which can never be fulfilled, and letting FindMinimum continue until it cannot find a sufficient decrease in the function, should be a way to find the best possible estimate of θ^*. This is because FindMinimum will then continue to minimize the objective function until it cannot do any better, instead of stopping because of a termination condition.

The objective function we use can usually not be evaluated for all possible values of the parameter vector θ. For example, some values of θ might cause the solutions to the differential equations in the prediction steps of the filters to explode (see e.g. Equation (2.42)). It is thus crucial that the steps taken by FindMinimum does not get outside the region where the objective function is possible to evaluate. A complicating matter is that, to our experience, FindMinimum sometimes performs function evaluations far away from the steps taken as part of the line search. We avoid these problems by supplying FindMinimum with the vectors θ_{min} and θ_{max}, specifying that θ should be in the range $\theta_{min} \leq \theta \leq \theta_{max}$. This vector inequality should be interpreted to hold for each component of θ. According to the Mathematica documentation [20], using these limits causes the optimization to stop whenever θ gets outside the specified range. However, it is our experience that FindMinimum will also not perform any function evaluations outside this range. This is ideal for us since we want to avoid function evaluations where the objective function cannot be evaluated. Note that problems can occur if the search reaches the limits of the range, and the search direction points outside the region. The search will stop with an error message in that case.

4 Differentiating the objective function of the parameter estimation framework

At the heart of the parameter estimation framework described in Section 3, is the minimization of the objective function $V(\theta; \mathscr{Y}_N)$ with respect to θ, as stated in Equation (3.2). $V(\theta; \mathscr{Y}_N)$ is defined in Equation (3.1) as $V(\theta; \mathscr{Y}_N)) \equiv -\log(L(\theta; \mathscr{Y}_N))$, where $L(\theta; \mathscr{Y}_N)$ is the likelihood function. The minimization is performed using a gradient based search method, and we calculate the gradient analytically. The first step in computing the analytical gradient of $V(\theta; \mathscr{Y}_N)$ is taken in Section 4.1, where the gradient of $V(\theta; \mathscr{Y}_N)$ is computed in terms of quantities that can be retrieved from the CD-UKF and the differentiated CD-UKF. The second step is thus naturally to differentiate the CD-UKF, and this is done in Section 4.2.

4.1 Gradient of the likelihood function

The gradient of the likelihood function has previously been computed in [16]. The lth component of the gradient of $V(\theta; \mathscr{Y}_N)$ can be computed as follows:

$$\frac{dV(\theta; \mathscr{Y}_N)}{d\theta_l} = \frac{1}{2}\sum_{k=1}^{N}\left(\frac{d}{d\theta_l}(\log(\det(S_k)) + \frac{d}{d\theta_l}(\epsilon_k^T S_k^{-1}\epsilon_k)\right), \tag{4.1}$$

where

$$\frac{d}{d\theta_l}(\log(\det(S_k)) = \frac{1}{\det(S_k)} \cdot \frac{d}{d\theta_l}(\det(S_k)) = \mathrm{Tr}\left(S_k^{-1}\frac{dS_k}{d\theta_l}\right) \tag{4.2}$$

and

$$\frac{d}{d\theta_l}(\epsilon_k^T S_k^{-1}\epsilon_k) = \frac{d\epsilon_k^T}{d\theta_l}S_k^{-1}\epsilon_k + \epsilon_k^T\frac{d}{d\theta_l}\left(S_k^{-1}\right)\epsilon_k + \epsilon_k^T S_k^{-1}\frac{d\epsilon_k}{d\theta_l}, \tag{4.3}$$

where

$$\frac{d}{d\theta_l}\left(S_k^{-1}\right) = -S_k^{-1}\frac{dS_k}{d\theta_l}S_k^{-1} \tag{4.4}$$

The derivative of a matrix determinant with respect to a scalar, as in (4.2), is given in [10]:

$$\frac{d}{d\theta_l}(\det(S_k)) = \det(S_k) \cdot \mathrm{Tr}\left(S_k^{-1}\frac{dS_k}{d\theta_l}\right),$$

where Tr denotes matrix trace. The derivative of the inverse of a matrix used in (4.4) is also given in [10].

The quantities necessary to compute $\frac{dV(\theta;\mathscr{Y}_N)}{d\theta_l}$ are ϵ_k, S_k, $\frac{d\epsilon_k}{d\theta_l}$ and $\frac{dS_k}{d\theta_l}$, where $\epsilon_k = y_k - \mu_k$ and $\frac{d\epsilon_k}{d\theta_l} = -\frac{d\mu_k}{d\theta_l}$. μ_k and S_k are calculated for $k = 1, .., N$ by running the CD-UKF. Analogously, $\frac{d\mu_k}{d\theta_l}$ and $\frac{dS_k}{d\theta_l}$ are calculated by running the differentiated CD-UKF, in which all the filter equations have been differentiated with respect to θ_l. (An explanation as to why it is necessary to differentiate all the filter equations is given in the beginning of Section 4.2)

4.1.1 Multiple measurement series

The objective function $V(\theta; \mathscr{Y}_N^1, .., \mathscr{Y}_N^s)$ using multiple measurement series is just

$$\sum_{i=1}^{s} V(\theta; \mathscr{Y}_N^i),$$

where $V(\theta; \mathscr{Y}_N^i)$ are the objective functions for the individual measurement series. This is described in Section 2.3.2. The derivative of $V(\theta; \mathscr{Y}_N^1, .., \mathscr{Y}_N^s)$ is thus

$$\sum_{i=1}^{s} \frac{dV(\theta; \mathscr{Y}_N^i)}{d\theta_l},$$

since differentiation is a linear operation.

4.2 Differentiated CD-UKF

The differentiated CD-UKF is run by providing the initial conditions $\frac{dm_0}{d\theta_l}$ and $\frac{dP_0}{d\theta_l}$, $l = 1, .., p$, and iterating the scheme through the prediction and update equations (4.5)-(4.20). $\frac{dm_0(\theta)}{d\theta_l}$ and $\frac{dP_0(\theta)}{d\theta_l}$ are calculated by simply performing the derivatives of $m_0(\theta)$ and $P_0(\theta)$ with respect to θ_l.

As discussed at the end of Section 4.1, the reason for differentiating the CD-UKF is to be able to calculate $\frac{d\mu_k}{d\theta_l}$ and $\frac{dS_k}{d\theta_l}$, as they are needed to compute $\frac{dV(\theta;\mathscr{Y}_N)}{d\theta_l}$. Is it not possible to simply differentiate the filter equations for μ_k and S_k (Equation (2.46) and (2.47)) and not the whole set of filter equations? The differentiation of (2.46) and (2.47) in (4.15) and (4.16), respectively, shows that $\frac{d\mu_k}{d\theta_l}$ and $\frac{dS_k}{d\theta_l}$ depends on $\frac{dY^-}{d\theta_l}$. This in turn depends on $\frac{dX^-}{d\theta_l}$ (Equation (4.14)). Continuing in the same manner leads to the conclusion that in order to calculate for example $\frac{d\mu_k}{d\theta_l}$ for $k = 4$, it is necessary to start from $\frac{dm_0}{d\theta_l}$ and $\frac{dP_0}{d\theta_l}$ and go through all of the differentiated filter equations from $k = 1$ to $k = 4$.

For simplicity, time dependent quantities such as $L(t)$ will in this section simply be denoted L. Also, the input signal u_t does not depend on the parameters θ, and will not be written out (so that $f(x_t, u_t, t, \theta)$ and $f_M(X_t, u_t, t, \theta)$ will be written $f(x, \theta)$ and $f_M(X, \theta)$, respectively).

$\Phi(\cdot)$ takes a matrix M and is defined by

$$\Phi_{ij}(M) = \begin{cases} M_{ij} & \text{if } i > j \\ \frac{1}{2}M_{ij} & \text{if } i = j \\ 0 & \text{if } i < j, \end{cases}$$

which results in a lower triangular matrix.

- *Prediction step*

  The derivative of (2.42) with respect to θ_l is

$$\frac{d}{d\theta_l}\left(\frac{dm}{dt}\right) = \frac{d}{dt}\left(\frac{dm}{d\theta_l}\right) = \frac{df_M(X,\theta)}{d\theta_l}w_m. \tag{4.5}$$

  The ith column of $f_M(X,\theta)$, denoted $(f_M(X,\theta))_i$, is $f(X_i,\theta)$ (see Equations (2.30) and (2.42)), so the ith column of $\frac{df_M(X,\theta)}{d\theta_l}$ is

$$\frac{d(f_M(X,\theta))_i}{d\theta_l} = \frac{df(X_i,\theta)}{d\theta_l} = \frac{\partial f(X_i,\theta)}{\partial X_i}\frac{dX_i}{d\theta_l} + \frac{\partial f(X_i,\theta)}{\partial\theta_l}. \tag{4.6}$$

  X is defined in (2.31):

$$X = \left[\begin{array}{ccc} m & ... & m \end{array}\right] + \sqrt{c}\left[\begin{array}{ccc} 0 & A & -A \end{array}\right], \tag{4.7}$$

  so $\frac{dX}{d\theta_l}$ is

$$\frac{dX}{d\theta_l} = \left[\begin{array}{ccc} \frac{dm}{d\theta_l} & ... & \frac{dm}{d\theta_l} \end{array}\right] + \sqrt{c}\left[\begin{array}{ccc} 0 & \frac{dA}{d\theta_l} & -\frac{dA}{d\theta_l} \end{array}\right]. \tag{4.8}$$

  In (2.28), A is defined as the lower triangular factor of the Cholesky decomposition of P. In [15], the following relation between the derivative of A and the derivative of P is derived:

$$\frac{dA}{dt} = A\Phi\left(A^{-1}\frac{dP}{dt}A^{-T}\right). \tag{4.9}$$

  t in this relation only plays the role of a scalar variable, so we can replace it with θ_l, which gives the relation

$$\frac{dA}{d\theta_l} = A\Phi\left(A^{-1}\frac{dP}{d\theta_l}A^{-T}\right). \tag{4.10}$$

  The derivative of (2.43) is

$$\frac{d}{d\theta_l}\left(\frac{dP}{dt}\right) = \frac{d}{dt}\left(\frac{dP}{d\theta_l}\right) = \frac{dX}{d\theta_l}Wf_M^T(X,\theta) + XW\frac{d\left(f_M^T(X,\theta)\right)}{d\theta_l} + \tag{4.11}$$
$$\frac{d\left(f_M(X,\theta)\right)}{d\theta_l}WX^T + f_M(X,\theta)W\frac{dX^T}{d\theta_l} + \frac{\partial L(\theta)}{\partial\theta_l}L(\theta)^T + L(\theta)\frac{\partial L(\theta)^T}{\partial\theta_l},$$

  where $\frac{dX}{d\theta_l}$ and $\frac{d(f_M(X,\theta))}{d\theta_l}$ have been computed in (4.8) and (4.6), respectively.

- *Update step*

  The derivative of (2.44) with respect to θ_l is

$$\frac{dX^-}{d\theta_l} = \left[\begin{array}{ccc} \frac{dm^-}{d\theta_l} & ... & \frac{dm^-}{d\theta_l} \end{array}\right] + \sqrt{c}\left[\begin{array}{ccc} 0 & \frac{dA^-}{d\theta_l} & -\frac{dA^-}{d\theta_l} \end{array}\right]. \tag{4.12}$$

The derivative of (2.45) is

$$\frac{dY^-}{d\theta_l} = \frac{dh_M(X^-, \theta)}{d\theta_l}, \tag{4.13}$$

where the ith column of $\frac{dh_M(X^-,\theta)}{d\theta_l}$ is given by

$$\frac{\partial h(X_i^-, \theta)}{\partial X_i^-} \frac{dX_i^-}{d\theta_l} + \frac{\partial h(X_i^-, \theta)}{\partial \theta_l} \tag{4.14}$$

analogous to (4.6). Furthermore, the derivatives of (2.46)-(2.51) are

$$\frac{d\mu}{d\theta_l} = \frac{dY^-}{d\theta_l} w_m, \tag{4.15}$$

$$\frac{dS}{d\theta_l} = \frac{dY^-}{d\theta_l} W (Y^-)^T + Y^- W \frac{d(Y^-)^T}{d\theta_l} + \frac{\partial R(\theta)}{\partial \theta_l}, \tag{4.16}$$

$$\frac{dC}{d\theta_l} = \frac{dX^-}{d\theta_l} W (Y^-)^T + X^- W \frac{d(Y^-)^T}{d\theta_l}, \tag{4.17}$$

$$\frac{dK}{d\theta_l} = \frac{dC}{d\theta_l} S^{-1} - C S^{-1} \frac{dS}{d\theta_l} S^{-1}, \tag{4.18}$$

$$\frac{dm}{d\theta_l} = \frac{dm^-}{d\theta_l} + \frac{dK}{d\theta_l}(y - \mu) - K \frac{d\mu}{d\theta_l} \tag{4.19}$$

and

$$\frac{dP}{d\theta_l} = \frac{dP^-}{d\theta_l} - \frac{dK}{d\theta_l} S K^T - K \frac{dS}{d\theta_l} K^T - K S \frac{dK^T}{d\theta_l}, \tag{4.20}$$

respectively.

5 Benchmarking

5.1 The logistic equation

The main purpose of using this benchmark model is to verify that the framework for parameter estimation is correct. The measurements used are generated from the logistic equation (Equation (5.1) and (5.2)). It is thus not real experimental data, but instead simulated data. To validate the analytical gradient of the CD-UKF, we will compare with the numerical gradient in Section 5.1.2. In that section we also compare with CD-EKF. It is not expected to give exactly the same results (since it is a different algorithm), but if it gets close, that is a good indication. In Section 5.1.3 we will create measurements without noise, to see if we can get close to the nominal parameter values.

We will also investigate some more properties of the framework in Section 5.1.4, where we plot the likelihood functions and look at the state-estimates using found parameter estimates and starting guesses of parameter estimates.

5.1.1 The system equations

The logistic equation describes the population growth in an ecological system. The state x_t is the population size at time t. The rate of reproduction is proportional to both the existing population and the amount of available resources. For a small population the growth rate is approximately exponential with growth factor a. As the population size increases, the limited resources causes the growth rate to decrease and approaches zero at the steady state population b.

$$dx_t = ax_t\left(1 - \frac{1}{b}x_t\right) + Ldw_t \tag{5.1}$$

$$y_k = x_k + v_k, \tag{5.2}$$

where v_k has variance R.

5.1.2 Comparison with numerical gradient and CD-EKF

The structural parameters a and b were estimated 100 times with each of the following methods:

- Using the CD-UKF with analytical gradient.
- Using the CD-UKF with a numerical gradient based on first order finite differences (see Section 2.5.1).
- Using the CD-UKF with a numerical gradient based on second order finite differences.
- Using the CD-EKF with analytical gradient.

Experimental parameters The parameter values used to generate the measurements were $a = 1$, $b = 2$, $L = 0.05$, $R = 0.004$, $x_0 = 0.2$ and $t_k = 0.16, 0.32, ..., 8$. Note that the process noise variance is $L^2 = 0.0025$, and the measurement noise variance is $R = 0.004$. The measurements for one realization are shown in Figure 3.

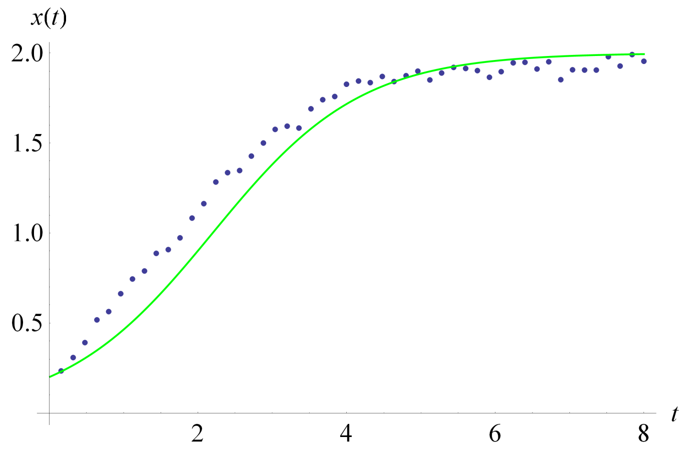


Figure 3: Simulated measurements for one realization of the logistic equation (dots), using process noise and measurement noise. The noise free system is also included (line). For this particular measurement series, the ML parameter estimates were $\hat{a} = 1.099$ and $\hat{b} = 1.935$, whereas the nominal values were $a = 1$ and $b = 2$. Thus, the parameter estimates are not very close to the nominal values. From the plot it is seen that the steady state value for the measurement series ($\hat{b}$) is lower than the steady state value of the noise free curve b, and that the measurement series is growing faster (controlled by the growth factor a) than the noise free curve. Thus, it seems reasonable that the ML method proposed $\hat{b}$ to be smaller than b, and $\hat{a}$ to be larger than a.

Method parameters The parameter values used in the filters were $L = 0.05$, $R = 0.004$, $m_0 = 0.2$ and $P_0 = 10^{-6}$. The value of P_0 was chosen by computing the likelihood for a few different values of P_0, using nominal values of a and b, and choosing a P_0 that gives a low value of the likelihood. The limits θ_{min} and θ_{max} used in the minimization (see Section 3.2.2) were 0.1 times the nominal values and 10 times the nominal values, respectively. The starting guesses of the parameters were drawn uniformly on the interval 0.5-2 times the nominal values. The termination condition was specified by letting the AccuracyGoal option in FindMinimum have the value ∞ (see Section 3.2.2). For the CD-UKF, the additional method parameters (compared to the CD-EKF) α, κ and β were set to 1, 0 and 2, respectively.

Results The results are presented in Table 1. θ_{nom} are the nominal, or true, values of θ; $\hat{\theta}_{mean}$ is the sample mean of the estimates of θ; $\hat{\theta}_{std}$ is the sample standard deviation of the estimates of θ; V^{min} is the sample mean of the value of V at the minimum; *Time* is the average total time spent on the optimization; *Steps* is the average number of steps taken by the minimization algorithm.

Table 1: Results with noise for 100 estimations of the parameters

Method	θ_{nom}	$\hat{\theta}_{mean}$	$\hat{\theta}_{std}$	V^{min}	Time	Steps
CD-UKF$_{\text{analytical}}$	$a = 1$	1.012	0.07895	-150.7	39.04	11.21
	$b = 2$	1.999	0.03117			
CD-UKF$_{\text{1st numerical}}$	$a = 1$	1.012	0.07890	-150.7	52.30	11.92
	$b = 2$	1.999	0.03114			
CD-UKF$_{\text{2nd numerical}}$	$a = 1$	1.012	0.07895	-150.7	40.69	11.63
	$b = 2$	1.999	0.03117			
CD-EKF$_{\text{analytical}}$	$a = 1$	1.011	0.07891	-150.7	9.532	11.14
	$b = 2$	1.999	0.03117			

The parameter estimates are essentially the same for the four settings. One purpose of these simulations was to verify that the analytical gradient of the CD-UKF is correct. It is therefore reassuring that the first order numerical gradient gives results close to the analytical gradient, and that the second order numerical gradient is even closer.

The parameter estimates using CD-EKF are similar to those of the CD-UKF (in fact, slightly better in this example).

5.1.3 Parameter estimation without noise

In this section, the measurements were generated without any system noise, i.e. neither process noise nor measurement noise. We estimated the structural parameters a and b starting from 100 different, randomly chosen, starting guesses drawn uniformly on the interval 0.5-2 times the nominal values.

The measurements were created without noise, but we cannot in the same way let the noise levels in the filters be zero. Doing so would force the state covariance P towards zero, since the filters would be certain about the system state after each measurement. If P is zero, the Cholesky decomposition used in the CD-UKF fails, since it requires a positive definite matrix. For these reasons, we use small positive values for the noise parameters in the filters.

Since the measurements do not contain any noise, they are deterministic, and there is only one possible measurement series for the given system to try the parameter estimation framework on. The parameter estimation framework is deterministic by construction, and therefore using the same measurement series many times will produce the same likelihood function. When using different starting guesses for the parameters, we should end up in approximately the same point if the likelihood function has only one minimum in the region covered by the starting guesses.

Experimental parameters The parameter values used to generate the measurements were $a = 1$, $b = 2$, $L = 0$, $R = 0$, $x_0 = 0.2$ and $t_k = 0.16, 0.32, ..., 8$. The measurements are shown in Figure 4.

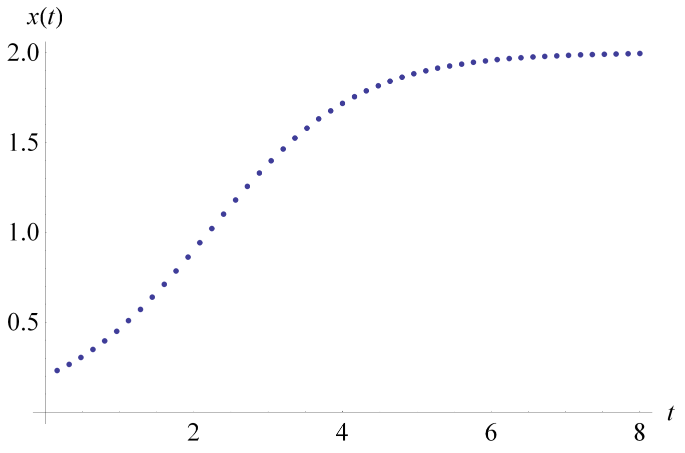


Figure 4: Noise-free measurements of the logistic equation.

Method parameters The method parameters used was the same as in Section 5.1.2 except that now L, R and P_0 were given the values 10^{-5}, 10^{-14} and 10^{-10}, respectively.

Results The results are presented in Table 2. The low values of $\hat{\theta}_{std}$ implies that the parameter estimation found the nominal parameters with good accuracy each time.

Table 2: Results without noise for 100 estimations of the parameters

Method	θ_{nom}	$\hat{\theta}_{mean}$	$\hat{\theta}_{std}$	V^{min}	Time	Steps
CD-UKF$_{\text{analytical}}$	$a = 1$	1.000	$2.248 \cdot 10^{-8}$	-622.0	43.14	11.85
	$b = 2$	2.000	$9.322 \cdot 10^{-9}$			

5.1.4 Plots of likelihood functions and state estimates

The likelihood function using the noise-free measurements in Section 5.1.3 are shown in Figure 5. The figure also includes the steps taken by the minimization algorithm, using the starting guess $a = 0.9624$, $b = 3.751$. The likelihood looks quite flat in some directions. However, the parameter estimation in Section 5.1.3 converged to the nominal parameters for 100 starting guesses uniformly drawn from the interval $0.5 \leq a \leq 2$, $1 \leq b \leq 4$, indicating that there is in fact a unique minimum in the region covered by the starting guesses. A contour plot is shown in Figure 6, where it is clear that there is a minimum at the nominal values of the parameters. The steps taken by the minimization algorithm are again included.

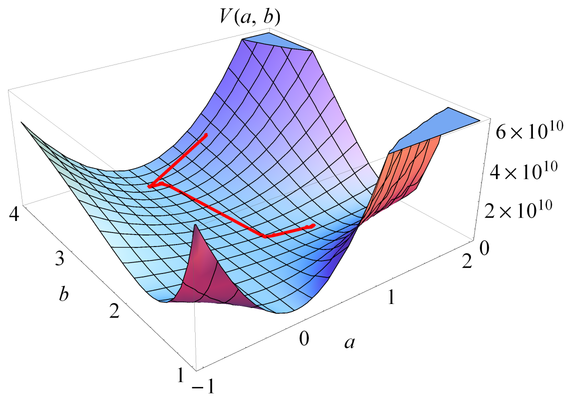


Figure 5: 3D Plot of the likelihood function for the noise free measurements. Included are the steps taken by the minimization algorithm for the starting guess $a = 0.9624$, $b = 3.751$. The minimum point was found to be at $a = 1.000$, $b = 2.000$.

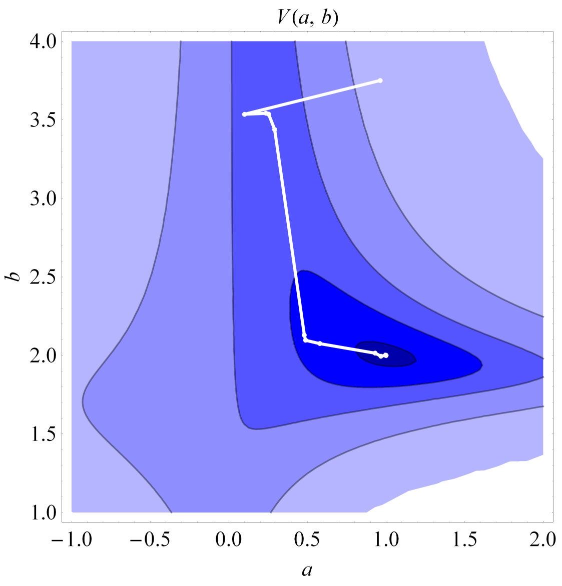


Figure 6: Contour plot of the likelihood function for the noise free measurements. Included are the steps taken by the minimization algorithm for the starting guess $a = 0.9624$, $b = 3.751$. The minimum point was found to be at $a = 1.000$, $b = 2.000$.

Figure 7 shows a plot of the likelihood function for the measurements with noise shown in Figure 3. Figure 8 shows a contour plot. Comparing with the noise free likelihood in Figure 5 and 6, we see that the noise makes the likelihood function more flat (compare the axes in Figure 5 and 7) and the minimum is slightly shifted away from the nominal parameters. The qualitative appearance is however the same.

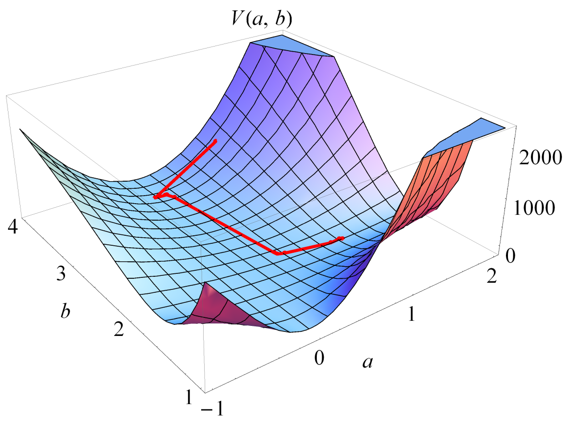


Figure 7: 3D Plot of the likelihood function for the measurements with noise. Included are the steps taken by the minimization algorithm for the starting guess $a = 0.9624$, $b = 3.751$. The minimum point was found to be at $a = 1.099$, $b = 1.935$.

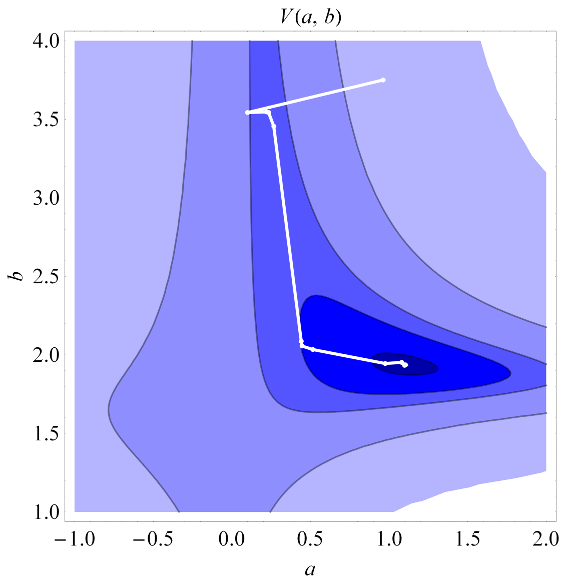


Figure 8: Contour plot of the likelihood function for the measurements with noise. Included are the steps taken by the minimization algorithm for the starting guess $a = 0.9624$, $b = 3.751$. The minimum point was found to be at $a = 1.099$, $b = 1.935$.

Figure 9 and 10 shows the state estimation using the starting guess and the found parameter estimates respectively, for the noise free measurements. Figure 11 and 12 shows the corresponding plots for the system with noise.

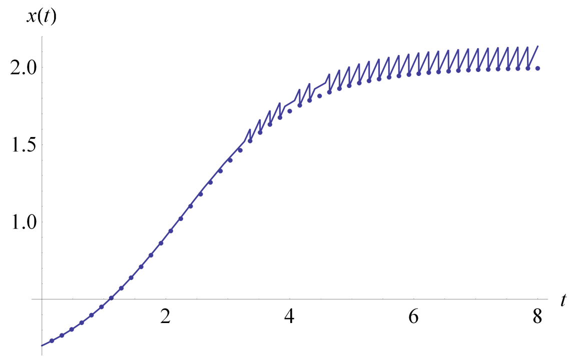


Figure 9: State estimation (line) with CD-UKF for the noise free measurements (dots) with the parameter starting guess $a = 0.9624$, $b = 3.751$. Notice that in the beginning, when x is small, the line is very close to the dots, but when the steady state phase is reached the state estimates are diverging between measurements, and pulled back at the measurement update. This is because for this particular starting guess, the growth factor a that dominates the dynamics for small x is close to the nominal value, but the steady state value b that dominates in the saturation phase is far from the nominal value.

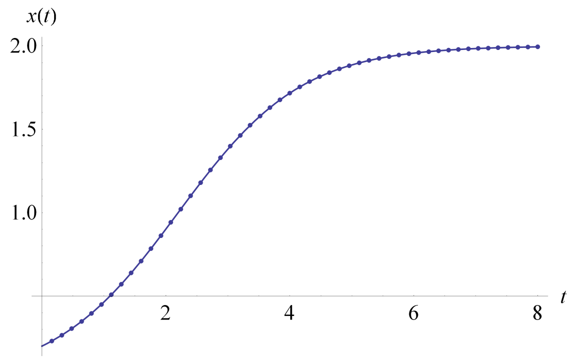


Figure 10: State estimation (line) with CD-UKF for the noise free measurements (dots), and with the ML parameter estimates $\hat{a} = 1.000$, $\hat{b} = 2.000$, which (almost) coincides with the nominal parameters.

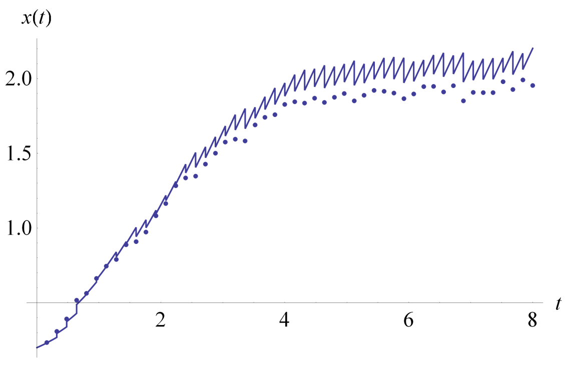


Figure 11: State estimation (line) with CD-UKF for a realization with noise (dots). It is the same realization as the one in Figure 3. The parameter starting guess $a = 0.9624$, $b = 3.751$ is used. In the steady state phase it is clearly seen that at the measurement update, the state estimate is pulled towards the measurements, but not all the way as in Figure 9. This is because in this Figure LL^T and R are of the same order of magnitude in the filters and thus, measurements and model are given approximately the same confidence. For the state estimation in Figure 9, LL^T is much bigger than R in the filters, i.e, the measurements are trusted more than the model predictions, and the state estimate is pulled all the way to the measurement.

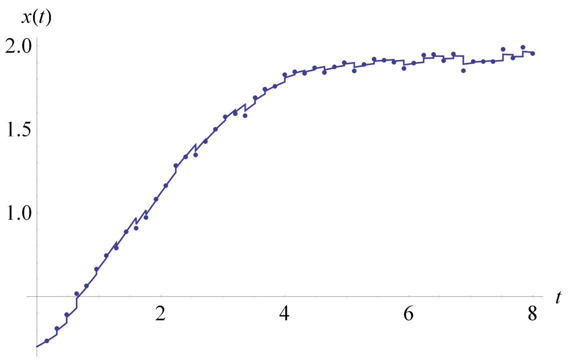


Figure 12: State estimation (line) with CD-UKF for the realization with noise (dots) shown in Figure 11. Now the ML parameter estimates $\hat{a} = 1.099$, $\hat{b} = 1.935$ are used.

5.2 Moles Mendes Banga biochemical pathway

This system is used to test the performance on a more challenging and realistic problem. This problem is studied in [9] and [13], but with a different approach, described in Section 5.2.1. Just like with the logistic equation, we compare the CD-UKF using analytical gradient with CD-UKF using numerical gradient and CD-EKF using analytical gradient.

5.2.1 The system equations

This is a dynamic model of a nonlinear biochemical pathway. It describes the variation of metabolite concentrations over time. The eight states, M_{1_t}, M_{2_t}, E_{1_t}, E_{2_t}, E_{3_t}, G_{1_t}, G_{2_t} and G_{3_t}, represents the concentrations of the species involved in the different biochemical reactions. S and P are initial concentrations of pathway substrate and pathway product, respectively. The equations contain 36 parameters [9].

$$\frac{dG_{1_t}}{dt} = \frac{V_1}{1+\left(\frac{P}{Ki_1}\right)^{ni_1}+\left(\frac{Ka_1}{S}\right)^{na_1}} - k_1 G_{1_t} \tag{5.3}$$

$$\frac{dG_{2_t}}{dt} = \frac{V_2}{1+\left(\frac{P}{Ki_2}\right)^{ni_2}+\left(\frac{Ka_2}{M_{1_t}}\right)^{na_2}} - k_2 G_{2_t} \tag{5.4}$$

$$\frac{dG_{3_t}}{dt} = \frac{V_3}{1+\left(\frac{P}{Ki_3}\right)^{ni_3}+\left(\frac{Ka_3}{M_{2_t}}\right)^{na_3}} - k_3 G_{3_t} \tag{5.5}$$

$$\frac{dE_{1_t}}{dt} = \frac{V_4 G_{1_t}}{K_4 + G_{1_t}} - k_4 E_{1_t} \tag{5.6}$$

$$\frac{dE_{2_t}}{dt} = \frac{V_5 G_{2_t}}{K_5 + G_{2_t}} - k_5 E_{2_t} \tag{5.7}$$

$$\frac{dE_{3_t}}{dt} = \frac{V_6 G_{3_t}}{K_6 + G_{3_t}} - k_6 E_{3_t} \tag{5.8}$$

$$\frac{dM_{1_t}}{dt} = \frac{\frac{kcat_1}{Km_1} E_{1_t}(S - M_{1_t})}{1+\frac{S}{Km_1}+\frac{M_{1_t}}{Km_2}} - \frac{\frac{kcat_2}{Km_3} E_{2_t}(M_{1_t} - M_{2_t})}{1+\frac{M_{1_t}}{Km_3}+\frac{M_{2_t}}{Km_4}} \tag{5.9}$$

$$\frac{dM_{2_t}}{dt} = \frac{\frac{kcat_2}{Km_3} E_{2_t}(M_{1_t} - M_{2_t})}{1+\frac{M_{1_t}}{Km_3}+\frac{M_{2_t}}{Km_4}} - \frac{\frac{kcat_3}{Km_5} E_{3_t}(M_{2_t}) - P}{1+\frac{M_{2_t}}{Km_5}+\frac{P}{Km_6}} \tag{5.10}$$

The paper [9] also attempts parameter estimation in this model. However, they approach the problem in a different way:

- They do not use a stochastic state space model (Section 2.1) to describe the process dynamics. Instead they use a white-box model, i.e. replace the SDE with an ODE, and take the output error (OE) approach discussed in Section 2.2.1 when estimating the parameters.
- They neither have process noise nor measurement noise when creating their measurement series. Thus, their measurements are completely deterministic, and if the nominal parameters are found, a perfect fit is achieved.
- As discussed in [9] and in section 2.2.1, the objective function corresponding to the OE problem formulation, is known for having many local optima, and one easily ends up in a local optimum if gradient based optimization is used. For this reason, they approach the problem using global optimization methods.

- They estimate all 36 structural parameters, whereas we only estimate 10.

5.2.2 Comparison with numerical gradient and CD-EKF

The measurements used have been generated by simulating (5.3) - (5.10), but with added process noise to all states. The outputs are simply all the state variables with added measurement noise. Sixteen measurement series are used with different combinations of values for S and P.

The reason for using several measurement series with different values for S and P, and not just one series, is that a larger set of dynamics is covered. Thus, the model is more thoroughly examined, and hence, the system dynamics is more easily revealed. This makes it easier to estimate the parameters (the likelihood function will be less flat).

The objective function using multiple measurement series $V(\theta; \mathscr{Y}_N^1, .., \mathscr{Y}_N^s)$, presented in Section 2.3.2, is a sum of the objective functions $V(\theta; \mathscr{Y}_N^i)$ for the individual measurement series $\mathscr{Y}_N^i$, $i = 1, .., s$ (in this case, $s = 16$). When computing the objective function for a particular measurement series, the particular values of S and P used to generate the measurement series is also used in the filter.

Experimental parameters The values of L and R are presented in Table 3. The different values for S and P are presented in Table 4. The four values of S and the four values of P are combined in the 16 possible unique ways, one for each measurement series. Apart from S and P, (5.3) - (5.10) contain 36 structural parameters. They have fixed values through the 16 measurement series, and are presented in Table 5 together with their numeric values. We attempt to estimate 10 of these parameters, presented in Table 7. For each of the 16 experiments, the model was simulated for 120 time units and measurements were taken at $t_k = 6, 12, ..., 120$, i.e. 20 measurements per experiment. The initial states were the same for all 16 experiments and are given in Table 6.

Table 3: The noise matrices L (process noise) and R (measurement noise) for the experiment generation, where I_n denotes the $n \times n$ identity matrix. NB: the values in L are standard deviations, but R is a covariance matrix. For ease of comparison against R, LL^T states the process noise covariance.

L	$10^{-3} I_8$
R	$4 \cdot 10^{-6} I_8$
LL^T	$10^{-6} I_8$

Table 4: The following values of S and P were combined in 16 ways, one for each measurement series

P	0.05	0.136	0.368	1.0
S	0.1	0.464	2.15	10

Table 5: Nominal parameter values for measurement generation

Parameter	Numeric value	Parameter	Numeric value	Parameter	Numeric value
V_1	1	V_3	1	V_6	0.1
Ki_1	1	Ki_3	1	K_6	1
ni_1	2	ni_3	2	k_6	0.1
Ka_1	1	Ka_3	1	$kcat_1$	1
na_1	2	na_3	2	Km_1	1
k_1	1	k_3	1	Km_2	1
V_2	1	V_4	0.1	$kcat_2$	1
Ki_2	1	K_4	1	Km_3	1
ni_2	2	k_4	0.1	Km_4	1
Ka_2	1	V_5	0.1	$kcat_3$	1
na_2	2	K_5	1	Km_5	1
k_2	1	k_5	0.1	Km_6	1

Table 6: Initial states used in all 16 experiments

G_{1_0}	0.66667	E_{2_0}	0.36409
G_{2_0}	0.57254	E_{3_0}	0.29457
G_{3_0}	0.41758	M_{1_0}	1.419
E_{1_0}	0.4	M_{2_0}	0.93464

Method parameters For the UKF, α, κ and β were set to 1, 0 and 2, respectively. In FindMinimum, the default values were used for AccuracyGoal and PrecisionGoal. In the filters, m_0 was chosen as the true initial state given in Table 6. P_0 was chosen as the process noise variance, which is $10^{-6}I_8$. L and R were chosen equal to the nominal values used for the experiment generation (see Table 3). Using the nominal noise parameter values in the filters means that the filters completely know the noise properties. Of course, when using real experiment data, the noise properties are seldom known. In that case, one approach is to estimate the noise parameters in the same way as the structural parameters are estimated. Another way is to fix them to some reasonable value.

Optimization results The 10 structural parameters, $kcat_1$, Km_1, Km_2, V_4, K_4, k_4, k_5, k_6, V_1 and Ki_1, were estimated once with the following methods:

- Using the CD-UKF with analytical gradient.
- Using the CD-UKF with a numerical gradient based on first order finite differences.
- Using the CD-UKF with a numerical gradient based on second order finite differences.

- Using the CD-EKF with analytical gradient.

The parameter starting guess, provided to the optimization routine, was drawn from a uniform distribution 0.5-2 times the nominal values and is shown in Table 7. The limits θ_{min} and θ_{max} was set to 10^{-12} and 10^{12}, respectively. The results of the optimization is shown in Table 8.

Table 7: Estimated parameters together with nominal values and starting guesses provided to the optimization routine

Parameter	Nominal value	Starting guess
$kcat_1$	1	1.545
Km_1	1	1.532
Km_2	1	1.503
V_4	0.1	0.1497
K_4	1	1.722
k_4	0.1	0.1768
k_5	0.1	0.1135
k_6	0.1	0.08143
V_1	1	1.885
Ki_1	1	1.782

Table 8: Results of optimization of Moles Mendes Banga pathway. The estimated parameters are the ones in Table 7 (in the same order)

Method	$\hat{\theta}$	V^{min}	Time	Steps
CD-UKF$_{\text{analytical}}$	0.9974 0.9998 1.009 0.09997 0.9959 0.1003 0.1001 0.1 0.9999 1.001	-13980	8 h 35 min	72
CD-UKF$_{\text{1st numerical}}$	0.9434 1.101 1.468 0.1205 1.674 0.08755 0.1001 0.1 0.9991 1.005	-12910	11 h 18 min	46
CD-UKF$_{\text{2nd numerical}}$	0.9974 0.9998 1.009 0.09998 0.996 0.1003 0.1001 0.1 0.9999 1.001	-13980	21 h 42 min	72
CD-EKF$_{\text{analytical}}$	0.9974 0.9999 1.009 0.09997 0.9959 0.1003 0.1001 0.1 0.9999 1.001	-13980	0 h 54 min	71

As can be seen in Table 8, the parameter estimates are in principle the same for all three methods, except the one using first order finite difference approximations of the gradient. Also, they are very close to the nominal parameter values, which are shown in Table 7. The value of the likelihood function is the same for these three methods with the presented number of digits. The method using first order finite differences reaches an estimate with a much worse likelihood value. Its proposed parameter estimate $\hat{\theta}$ is not as close to the nominal values as the parameter estimates produced by the other three methods. The number of steps taken is lowest for the method using first order differences. However, since it reaches a completely different solution, no conclusion can be drawn from this. For the two other UKF-based methods, the number of steps were the same (72), whereas the EKF stopped after 71 steps.

The path to the minimum is very similar for CD-UKF$_{\text{analytical}}$ and CD-UKF$_{\text{2nd numerical}}$, see Appendix A. They take the same number of steps and they end up in approximately the same estimate. This indicates that the second order numerical gradient approximates the analytical gradient very well. The poor result for the CD-UKF$_{\text{1st numerical}}$ on the other hand, indicates that the first order approximation does not. This is further discussed in Section 5.3.

Even though the CD-UKF$_{\text{2nd numerical}}$ took the same number of steps and reached practically the same parameter estimates, the computation time was approximately 2.5 times that of the CD-UKF$_{\text{analytical}}$.

For this particular example, both the CD-EKF and CD-UKF arrived practically at the same solution, but the CD-EKF was almost 10 times faster. Thus, for this example, there is no gain in using the CD-UKF over the CD-EKF.

For our first benchmark example, the logistic equation (see Section 5.1), we performed 100 optimization runs, and could therefore draw conclusions from a statistical point of view. For this bigger example, the computational time is much larger, and thus we only performed one run.

5.3 Exploration of the bad results using first order finite differences when computing the gradient

In the previous sections where the benchmark results were presented, it was noted that the optimization result was different when using CD-UKF with first order finite differences approximation of the gradient, compared to the case with analytic gradient and second order difference approximation. This was very clear in the case of the Moles Mendes Banga biochemical pathway in Section 5.2. In Section 5.1, where the logistic equation was studied, this was not as evident, but the CD-UKF$_{\text{1st numerical}}$ took more steps and considerably longer time compared to CD-UKF$_{\text{2nd numerical}}$.

The reason for the poor result using first order finite differences to approximate the gradient stems from Mathematica's NDSolve, that is used to solve the involved ODEs. When NDSolve numerically integrates the ODE

$$\frac{dx_t}{dt} = f(x_t, u_t, t, \theta) \tag{5.11}$$

it uses adaptive step length. If (5.11) is solved with NDSolve for a particular parameter vector θ, and then solved again, but with a very small change in θ, then by continuity, a very small change is expected in the solution to the ODE as well (assuming no bifurcation). However, a small change in θ can cause NDSolve to take a different set of steps when solving the ODE, leading to a (relatively) large change in the solution. This will introduce spikes in the objective function $V(\theta; \mathscr{Y}_N)$ (see Figure 13), which introduces errors in the finite differences used to approximate the gradient. The spikes can be thought of as numerical noise.

Truncation error and rounding error In Section 2.5.1 finite difference approximations of derivatives were introduced. The first order approximation of the derivative of a single variable function $f(x)$ is

$$\frac{f(x+h)-f(x)}{h} = f'(x) + \mathcal{O}(h), \tag{5.12}$$

where h is the step size. There are two types of errors associated with using this formula to approximate $f'(x)$ [2]: *truncation error* and *rounding error*. The truncation error is captured by the $\mathcal{O}(h)$-term. The rounding error is caused by errors in the function values, caused by the finite precision in evaluating $f(x)$. If the error in $f(x)$ is bounded by ϵ, the rounding error in evaluating the left hand side of (5.12) is bounded by $2\epsilon/h$. The total computational error can now be expressed as the sum of the truncation error and rounding error:

$$\frac{2\epsilon}{h} + \mathcal{O}(h). \tag{5.13}$$

Decreasing the step size h will decrease the truncation error, but increase the rounding error.

Logistic system Figure 13 shows a plot of the objective function for the logistic system using the settings with noise as in Section 5.1.2. The objective function is plotted around a randomly chosen point, denoted $a_0 = 0.9624$, $b_0 = 3.751$, in the a-direction. The value of the objective function in that point is $V(a_0, b_0) \equiv V_0 = 568.7$. We can see that there is a noise-like feature in the objective function. The noise is approximately on the order of 10^{-7}. The noise looks rather random and uncorrelated, so a reasonable approach is to consider the noise to cause a decrease in the precision of the objective function, as modeled by ϵ above. In this case, we thus let ϵ be of the order 10^{-7}, $\epsilon \sim 10^{-7}$.

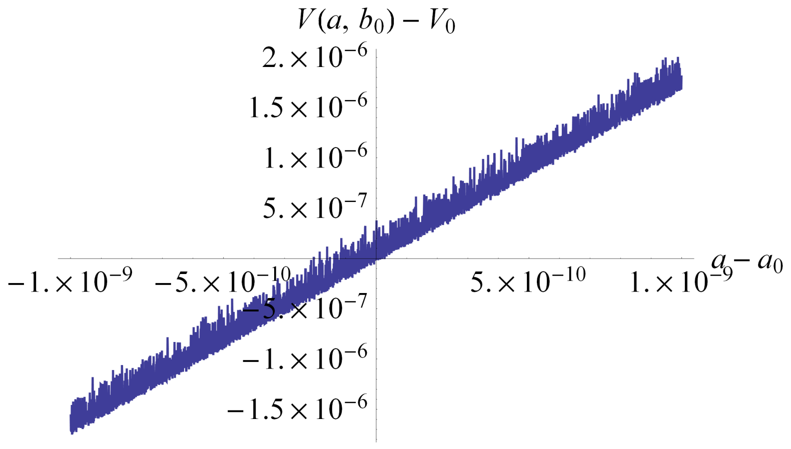


Figure 13: Objective function for the logistic system. The noise is due to adaptive step length in the numerically solved ODEs.

The noisy appearance of the objective function may or may not introduce significant errors in the finite difference approximations of derivatives. Referring to (5.12) and (5.13), the rounding error is significant if

$$\frac{2\epsilon}{h} \gtrsim f'(x). \tag{5.14}$$

Looking at a few examples, we have seen that when Mathematica uses first order differences, $h \sim 10^{-8}$; when it uses a second order scheme, $h \sim 10^{-5}$. With a first order scheme and $\epsilon \sim 10^{-7}$ (as in Figure 13), the error will then be significant if $f'(x) \lesssim 10$. In Figure 13, $\frac{\partial V(a,b_0)}{\partial a} \sim 1000$, so the rounding error is not a problem in that case. That is also clear from the figure, since the noise is relatively small (compared to the slope of the curve) already when a is varied on a 10^{-9}-scale.

To convince ourselves that the noise in the objective function is caused by the adaptive step length used in NDSolve, we will reproduce Figure 13, but using a fixed step length in NDSolve. This is done in Figure 14. The value of the objective function is now slightly different, $V_0 = 568.6$. We used the explicit Euler method with a fixed step size of 1/1000.

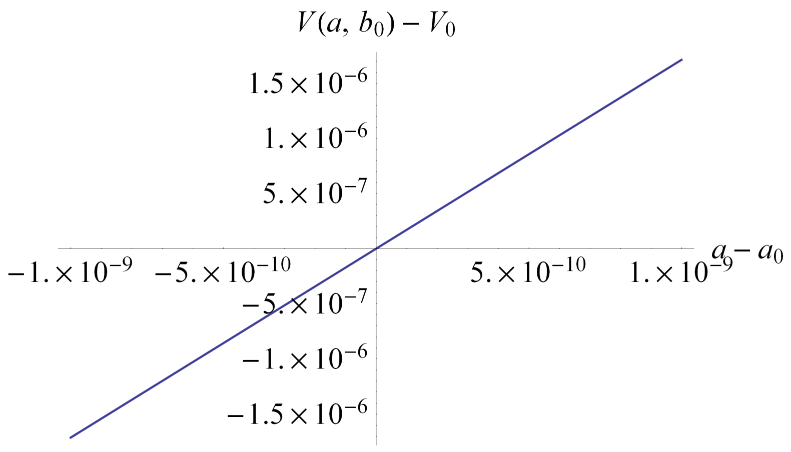


Figure 14: Objective function of the logistic system using fixed step length in the numerically solved ODEs. There is no visible noise in the objective function when a varies over this scale.

We can see that the objective function no longer appears noisy on this scale. Since the computations are done with finite precision, the objective function will of course be noisy, or at least quantized, on some scale. We investigate when this effect occurs when using fixed step length in Figure 15. The uncertainty in the objective function (ϵ) is in this case on the order of 10^{-12}.

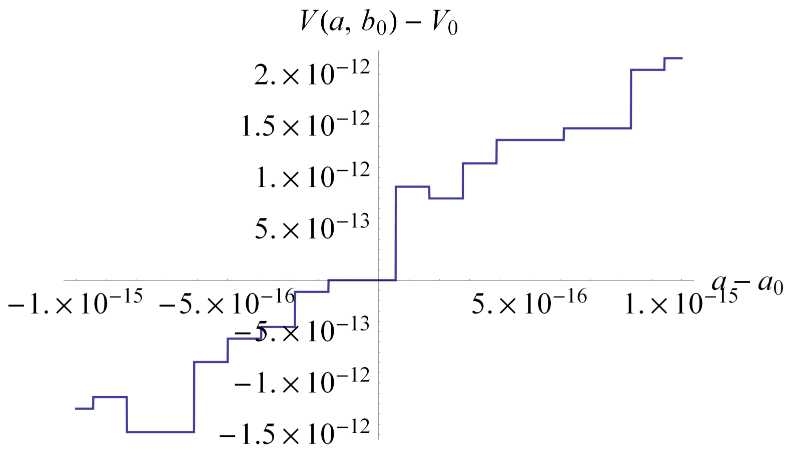


Figure 15: Objective function of the logistic system using fixed step length in the numerically solved ODEs. a now varies on the working precision scale ($\approx 10^{-16}$), and the "noise" in the objective function is on the order of 10^{-12}.

Moles Mendes Banga biochemical pathway Using this system, the optimization using first order finite differences (CD-UKF$_{\text{1st numerical}}$) arrived at a completely different solution than those using second order finite differences (CD-UKF$_{\text{2nd numerical}}$) and analytical gradient (CD-UKF$_{\text{analytical}}$). Comparing the steps taken by the optimization algorithms (see Appendix A), the steps taken by the CD-UKF$_{\text{1st numerical}}$ takes a different path (but not very different) the first 35-40 steps or so, compared with CD-UKF$_{\text{2nd numerical}}$ and CD-UKF$_{\text{analytical}}$. In the following steps, CD-UKF$_{\text{2nd numerical}}$ and CD-UKF$_{\text{analytical}}$ considerably improve the estimates, but CD-UKF$_{\text{1st numerical}}$ gets stuck and fails to improve the estimates. The steps taken by CD-UKF$_{\text{2nd numerical}}$ and CD-UKF$_{\text{analytical}}$ are quite similar to each other through the whole optimization.

We will now show that the noise in the objective function causes significant errors in the gradient when using first order finite differences. The first component of the gradient, $\frac{\partial V}{\partial kcat_1}$, was computed with the settings used in Section 5.2 and with the θ-parameters set to the starting guess that was used in the parameter estimation in Section 5.2. $\frac{\partial V}{\partial kcat_1}$ was computed using analytical gradient, as well as first and second order finite differences; the results were 836700, 1.550×10^6 and 836600, respectively. The error is thus on the order of 10^6 using first order differences.

The step sizes used by FindMinimum in the optimization using first and second order finite differences were $h_1 = 2.302 \times 10^{-8}$ and $h_2 = 9.356 \times 10^{-6}$, respectively. If the error using first order differences is caused by the noisy objective function, the noise should be of the order $\epsilon \sim 10^{-2}$. Then $2\epsilon/h_1 \sim 10^{-2}/10^{-8} = 10^6$. Figure 16 shows the objective function, where noise appears, and it is on the order of 10^{-2}. The noisy objective function can thus explain the error in the first order finite difference approximation of $\frac{\partial V}{\partial kcat_1}$.

There is still the possibility that the error in the approximation of $\frac{\partial V}{\partial kcat_1}$ is caused by both rounding error from noise, and by truncation error. To exclude the second possibility, we compute $\frac{\partial V}{\partial kcat_1}$ using a larger step size h_1. This should increase the truncation error (Equation (5.13)), so if the approximation of $\frac{\partial V}{\partial kcat_1}$ becomes accurate, the truncation error is not significant. Using $h_1 = 10^{-4}$ should give rounding errors of the order $10^{-2}/10^{-4} = 10^2$. The value of $\frac{\partial V}{\partial kcat_1}$ using a first order difference with $h_1 = 10^{-4}$ was 836800, which is approximately as accurate as the second order difference approximation.

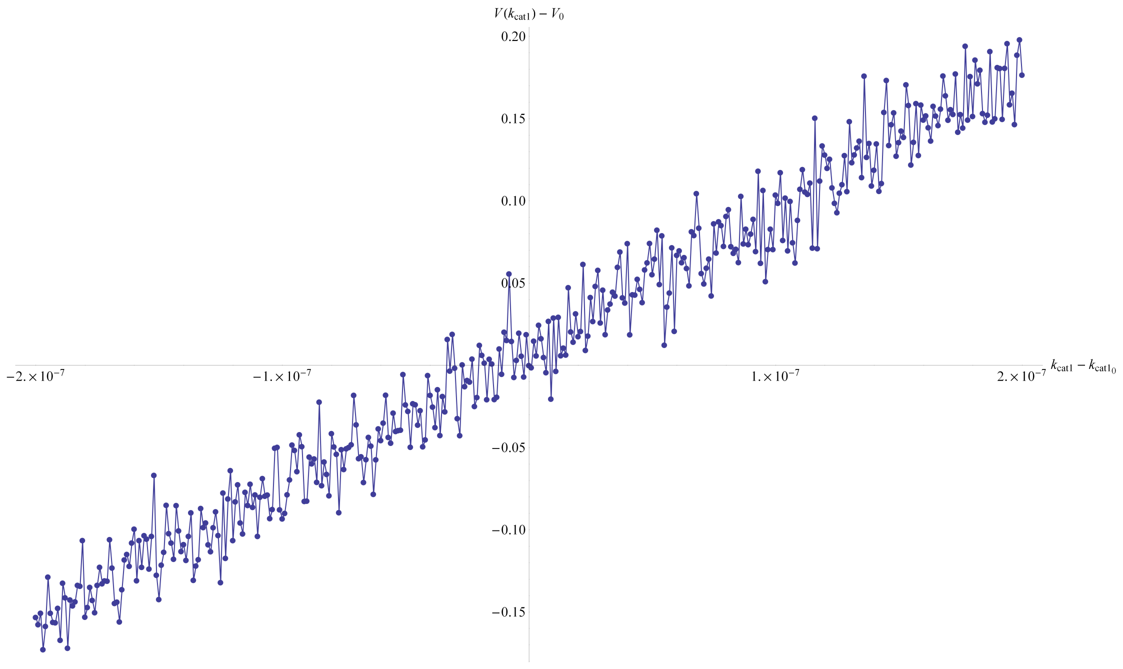


Figure 16: Objective function of the Moles Mendes Banga biochemical pathway from Section 5.3. The dots indicate where the objective function was evaluated (the objective function is quite heavy to compute, so we limited the number of points to evaluate it in). The objective function appears to be noisy on the order of 10^{-2}. The value of V_0, which is $V(kcat_{1_0})$, is 1.271×10^7.

6 Discussion

Comparison between CD-UKF and CD-EKF In Section 5 we compared the CD-UKF with the CD-EKF as state estimators in the parameter estimation framework, using two benchmark problems. We tested the CD-EKF and CD-UKF using the analytical gradient of the objective function, as well as two different numerical gradient approximations in the case of the CD-UKF. Using the analytical gradient with both CD-UKF and CD-EKF, a fair comparison between the filters can be made and it is discussed below.

As was discussed in Section 1, [17] conducted a similar comparison between the DD-UKF and the DD-EKF. They claimed that the DD-UKF gave better parameter estimates than the DD-EKF. As was further discussed in Section 1, both the DD-UKF and the CC-UKF provide better state estimation than their respective EKF. The authors to [17] inferred this as the reason why the DD-UKF gave better parameter estimates than the DD-EKF in their study. Therefore, we had reasons to believe that using the CD-UKF would improve the parameter estimates compared to using the CD-EKF as state estimator. In our tests however, we could not repeat the results of [17] for the CD-UKF. Instead the CD-EKF and CD-UKF gave in principle the same parameter estimates in both benchmark problems, but the CD-EKF needs significantly less computation time. According to [15], the CC-UKF is a better state estimator than the CC-EKF when the estimation uncertainties are significant, i.e. in case of large system noise. Thus, it is possible that the CD-UKF had surpassed the CD-EKF in our benchmarking section, had we used more system noise.

Comparison between analytical gradient and numerically approximated gradient A major part of our work has been to derive expressions for the analytical gradient of the objective function and implement it in Mathematica. As discussed above, in Section 5 we compared the analytical gradient with two different numerical gradient approximations in the case of the CD-UKF. The first gradient approximation used first order finite differences, and the second used second order finite differences. As shown in Section 5, using second order differences to approximate the gradient gave in principle the same parameter estimates as using the analytical gradient for both our benchmarking problems. For the problem in Section 5.2, the method using first order gradient approximations got a much worse result than the method using the analytical gradient. In Section 5.3, we showed that this was due to numerical errors caused by the fact that Mathematicas NDSolve uses adaptive step length when integrating ODEs. These errors introduced numerical noise on the objective function.

However, we have observed that in our experiments, when Mathematica's FindMinimum uses first order finite differences to approximate the objective function gradient, it uses a step length of the order 10^{-8} (see Section 5.3). This small step length gives, in the example treating the Moles Mendes Banga problem in Section 5.3, a significant rounding error. It is also shown in Section 5.3 that increasing the step size to 10^{-4} made the first order approximation of the gradient close to the analytical gradient, i.e. both the truncation error and rounding error were small. This suggests that, even though numerical errors causes the objective function to be spiky, by choosing a proper step length, first order differences can give a good approximation of the objective function gradient. But then the question of what actually is a good step length arises, and the answer may vary from problem to problem. In Section 5.1 it was seen that the optimization using first order differences needed more computation time than the optimization using second order differences. This is not expected based on the fact that approximating the gradient with first order differences requires less function evaluations than using a second order difference approximation, as discussed in Section 2.5.1. The cause of the larger computation time was probably the numerical noise introduced by NDSolve.

It should be noted that the investigation of the numerical gradient approximations in Section 5.3 was done far from the minimum of the objective function. When getting close to the minimum of the objective function, the norm of the gradient is small. To avoid too large rounding error compared to the true value of the gradient near the minimum, it may be necessary to increase the step length (see Equation (5.14)). However, increasing the step length will increase the truncation error instead. One approach to this issue can be to use first order differences in the beginning of the optimization and, as the norm of the gradient decreases, swap to second order differences. If one uses that approach one must confront the problem of deciding what is a proper step length and when to switch from first order to second order differences. If one instead uses the analytical gradient, no such decisions need to be taken.

Of course, one wants to arrive to the minimum as fast as possible, so an interesting question is: is the analytical gradient faster than a numerical approximation or not? In our benchmarking problems, the method using the analytical gradient was the fastest for both our problems. For the logistic equation, where two parameters were estimated, using second order differences was almost as fast as using the analytical gradient (see Table 1). Remember from Section 2.5.1 that a second order approximation of the gradient requires $2p$ objective function evaluations (where p is the number of parameters), whereas a first order approximation requires only $p + 1$ evaluations. Thus, if a proper step length had been used for the optimization using first order differences, it is expected that it would have required less computation time than the optimization using the analytical gradient.

On the Moles Mendes Banga problem, described in Section 5.2, where 10 parameters were estimated, the optimization using second order differences required approximately 2.5 times more computation time than using the analytical gradient. If the first order difference approximation had been as accurate as the second order approximation, the computation time of the optimization would have dropped slightly less than 50% compared with using second order differences. But this is still more compuation time than using analytical gradient. In Section 2.5.1 it was shown that the computation time for approximating the gradient with finite differences increases (approximately) linearly in the number of parameters. It may be that the computation time for the analytical gradient increases slower than linearly in the parameters, and would in that case be the fastest alternative when estimating many parameters. To say something with certainty on this matter, more tests must be done.

6.1 Possible future work

In this thesis we only estimated the structural parameters, i.e. parameters in $f(\cdot)$ and $h(\cdot)$ in Equations (2.1) and (2.2), respectively, and not the noise parameters, i.e. parameters in $L(\cdot)$ and $r_k(\cdot)$. Estimating the noise parameters is desirable because it provides a measure of the model uncertainty. It is possible to estimate the noise parameters (in the same manner as the structural parameters) within our implemented framework.

We wrote in the beginning of Section 6 that the UKF is a better state estimator than the EKF in case of large system noise, and that it is thus possible that the CD-UKF had surpassed the CD-EKF in our benchmarking section, had we used more system noise. This is rather straight forward to investigate, one can e.g take our benchmarking problems and repeat the parameter estimation but with more system noise.

It would be interesting to investigate how the computation time for the analytical gradient increases as a function of the number of parameters. This could e.g. be done using the Moles Mendes Banga problem. One could choose a point in the parameter space (or several) and calculate the analytical gradient in this point, first w.r.t. one parameter, then w.r.t. two parameters and so on. The computation time could then be plotted as a function of the number of estimated parameters. In the same plot one could also plot the same quantity for the finite differences approximated gradients, which are expected to increase linearly in the parameters. An interesting question is thus: does the computation time for the analytical gradient increases slower than linear in the parameters?

Another interesting issue to investigate would be to generate measurements by simulating one system model and then conduct parameter estimation in a simplified model. For example, consider simulating this nonlinear model of a pendulum:

$$dx_{1_t} = x_{2_t} dt$$

$$dx_{2_t} = -(a \sin(x_{1_t}) + b x_{2_t}) dt,$$

and creating measurements by sampling it at discrete time instants, possibly with added measurement noise. Then the parameters a and b could be estimated in the simpler, linear model:

$$dx_{1_t} = x_{2_t} dt$$

$$dx_{2_t} = -(a x_{1_t} + b x_{2_t}) dt + L(t, \theta) d\beta_t.$$

This is interesting because in reality, the real process is not known exactly and the model always contains simplifications. It is therefore interesting to see if good estimates of a and b could be achieved using the simplified model and also to estimate the noise parameters to provide a measure of the model uncertainty. One could also conduct the parameter estimation using a white box model, described in Section 2.2.1, and compare the results.

A Steps taken by the optimization algorithm on the Moles Mendes Banga biochemical pathway system in Section 5.2

Table 9: Steps taken using the method CD-UKF$_{\text{analytical}}$ (72 steps)

k_{cat1}	K_{m1}	K_{m2}	V_4	K_4	k_4	k_5	k_6	V_1	K_{i1}
1.51656	1.54339	1.49573	0.0440611	1.72675	0.244549	1.00001×10^{-12}	0.126862	1.01542	1.67273
1.5159	1.54356	1.49553	0.131066	1.72578	0.175252	0.115224	0.117604	0.841174	1.64962
1.51358	1.54443	1.49494	0.160572	1.72406	0.139545	0.0952009	0.106918	0.849169	1.64766
1.49215	1.55294	1.48934	0.157028	1.72349	0.126419	0.0986359	0.0649192	0.93664	1.64503
1.47221	1.56074	1.48414	0.155145	1.72372	0.120754	0.0975852	0.115441	0.971282	1.63866
1.46226	1.5645	1.48156	0.151153	1.72472	0.128421	0.09332	0.10034	0.930002	1.63063
1.4359	1.57451	1.47475	0.147076	1.72477	0.119077	0.0946422	0.0918363	0.927091	1.61774
1.32009	1.61809	1.44491	0.126402	1.72596	0.0918693	0.0975964	0.0804356	0.929944	1.56119
1.14357	1.6836	1.39965	0.0950498	1.7279	0.0604897	0.100969	0.0811472	0.93875	1.47089
1.09752	1.70032	1.38789	0.0869211	1.72808	0.0631907	0.102148	0.0915166	0.947504	1.43896
1.08802	1.70358	1.3854	0.0864713	1.72819	0.0603591	0.100759	0.102496	0.946121	1.41859
1.09553	1.70045	1.38728	0.087821	1.72811	0.0628923	0.100769	0.104846	0.949449	1.40617
1.10813	1.69322	1.39028	0.0910443	1.72806	0.068766	0.100697	0.112512	0.96575	1.30672
1.13041	1.6784	1.39552	0.0981762	1.72808	0.0781925	0.1006	0.120776	0.998414	1.05768
1.14446	1.67187	1.39912	0.101617	1.72801	0.0784449	0.10026	0.112964	1.00299	1.0287
1.14251	1.67086	1.39883	0.103196	1.72793	0.0740254	0.0996234	0.0975902	1.00085	1.01301
1.13611	1.67225	1.39724	0.102714	1.72792	0.0735921	0.0999494	0.0999821	1.00169	0.989248
1.13511	1.67262	1.39708	0.102543	1.72787	0.0732775	0.100003	0.100065	0.999438	1.00479
1.13532	1.67197	1.39723	0.102931	1.72783	0.0735536	0.0999413	0.0997366	0.999627	1.00338
1.13497	1.67044	1.39741	0.103821	1.72768	0.0741808	0.0999025	0.099494	0.999759	1.00185
1.1328	1.66421	1.39805	0.107365	1.72705	0.0766631	0.09984	0.0990247	0.999973	0.998999
1.12809	1.6528	1.3991	0.113608	1.72586	0.0810206	0.0998302	0.0987273	1.00007	0.997316
1.12035	1.63564	1.40058	0.122443	1.72405	0.0871788	0.0999205	0.0988722	0.999921	0.998456
1.11415	1.62257	1.40166	0.128278	1.72267	0.091249	0.100064	0.0994381	0.999593	1.00205
1.11114	1.61623	1.40219	0.129937	1.72204	0.0924143	0.100164	0.0999322	0.99934	1.00507
1.1082	1.60937	1.40281	0.130236	1.7214	0.0926379	0.100257	0.100426	0.999098	1.00807
1.10184	1.59264	1.40446	0.129789	1.71991	0.0923688	0.100392	0.10117	0.998737	1.01261
1.08628	1.54865	1.40897	0.128507	1.71606	0.0915944	0.100583	0.10226	0.9982	1.01936
1.04854	1.43728	1.42067	0.126191	1.70636	0.0902976	0.10083	0.103734	0.997441	1.02877
0.980741	1.23009	1.44281	0.122965	1.68838	0.0886549	0.100985	0.104816	0.996823	1.03616
0.941167	1.09998	1.45718	0.119573	1.67723	0.0866807	0.100751	0.103725	0.997391	1.02928
0.939097	1.08214	1.45961	0.115948	1.67589	0.0842038	0.100152	0.100555	0.99924	1.00772
0.94425	1.09941	1.45764	0.118545	1.67727	0.0859408	0.100045	0.0999205	0.999562	1.0038
0.943372	1.09564	1.45809	0.119664	1.67687	0.086769	0.100046	0.0999094	0.99954	1.00394
0.943157	1.0953	1.45809	0.120007	1.67677	0.0870358	0.10005	0.0999262	0.999528	1.00406
0.943241	1.09542	1.45805	0.120078	1.6767	0.0870948	0.100051	0.0999304	0.999524	1.00409
0.94348	1.09568	1.45777	0.120394	1.67607	0.0873681	0.100056	0.0999487	0.999512	1.00419
0.943899	1.09606	1.457	0.120832	1.67427	0.0877824	0.100065	0.0999756	0.999493	1.00434
0.94468	1.09651	1.4547	0.12152	1.66883	0.0885299	0.100078	0.100021	0.999463	1.0046
0.946165	1.09675	1.44839	0.122499	1.65371	0.0898711	0.100101	0.100095	0.999417	1.00499
0.949334	1.09576	1.43025	0.123849	1.60997	0.0925995	0.100141	0.100223	0.999345	1.00563
0.957877	1.08932	1.36978	0.125546	1.46366	0.099666	0.100224	0.100485	0.999224	1.00674
0.964648	1.08468	1.32178	0.124551	1.34785	0.103809	0.100272	0.100637	0.999193	1.00713
0.977645	1.08118	1.24163	0.117931	1.15597	0.10753	0.10034	0.10085	0.99923	1.00716
0.988516	1.08188	1.18505	0.108426	1.02157	0.106912	0.100372	0.100945	0.999358	1.00651
0.98711	1.05332	1.15255	0.100051	0.934218	0.105048	0.100193	0.100289	0.999841	1.00175
0.979293	1.03831	1.15642	0.103389	0.939274	0.10711	0.100082	0.0998981	0.999933	0.99995
0.973607	1.04439	1.18659	0.106147	1.01405	0.105623	0.100097	0.0999495	0.99981	1.00106
0.975742	1.04223	1.17026	0.103629	0.974563	0.10547	0.100123	0.10003	0.99982	1.00129
0.975253	1.03753	1.16514	0.102395	0.960838	0.104999	0.10012	0.100025	0.999854	1.00106
0.97486	1.03603	1.16299	0.100669	0.955543	0.103458	0.100114	0.100013	0.999885	1.0009
0.974682	1.0385	1.16777	0.0989329	0.968118	0.100821	0.100104	0.0999868	0.999905	1.00084
0.975674	1.04269	1.17024	0.0987567	0.975444	0.100224	0.100101	0.0999719	0.999903	1.00085
0.975872	1.04402	1.17217	0.0989596	0.980478	0.100164	0.100101	0.0999692	0.999897	1.00089
0.975831	1.04418	1.17252	0.099033	0.981396	0.100191	0.100101	0.0999697	0.999895	1.0009
0.975775	1.04412	1.17265	0.0990565	0.981753	0.100196	0.100101	0.0999701	0.999895	1.0009
0.975648	1.04397	1.17288	0.099105	0.982491	0.100205	0.100101	0.0999707	0.999894	1.00091
0.975479	1.04371	1.1731	0.0991725	0.983552	0.100215	0.100101	0.0999717	0.999892	1.00092
0.975228	1.04319	1.17314	0.0992872	0.985376	0.100232	0.100101	0.0999734	0.99989	1.00094
0.974933	1.04219	1.17241	0.0994692	0.988321	0.100256	0.100101	0.0999762	0.999886	1.00096
0.974726	1.04013	1.16915	0.0997617	0.993185	0.100286	0.100102	0.0999809	0.99988	1.00101
0.975088	1.03585	1.15898	0.100202	1.00086	0.100308	0.100105	0.0999886	0.99987	1.00109
0.977155	1.02763	1.13374	0.100749	1.01131	0.100279	0.10011	0.0999994	0.999856	1.0012
0.981856	1.01617	1.09207	0.10109	1.01964	0.100143	0.100117	0.100008	0.999847	1.00129
0.987554	1.00675	1.05362	0.100916	1.01878	0.0999836	0.100122	0.100008	0.99985	1.00129
0.993677	1.0018	1.0253	0.100266	1.00491	0.10007	0.100126	0.0999988	0.999866	1.00117
0.997422	1.00003	1.00813	0.0999746	0.994959	0.100333	0.100129	0.0999992	0.999871	1.00111
0.997134	1.00079	1.01226	0.0999164	0.994336	0.100299	0.100129	0.0999988	0.999871	1.00111
0.997248	1.00017	1.01011	0.099962	0.995764	0.100268	0.100129	0.0999992	0.99987	1.00112
0.997361	0.999958	1.00938	0.0999671	0.995836	0.10027	0.100129	0.0999993	0.99987	1.00112
0.997421	0.999833	1.00899	0.0999731	0.995936	0.100271	0.100129	0.0999995	0.99987	1.00112
0.997421	0.999832	1.00899	0.0999737	0.99594	0.100271	0.100129	0.0999995	0.99987	1.00112

Table 10: Steps taken using the method CD-UKF$_{\text{1st numerical}}$ (46 steps)

k_{cat1}	K_{m1}	K_{m2}	V_4	K_4	k_4	k_5	k_6	V_1	K_{i1}
1.48093	1.53518	1.49493	1.00001×10^{-12}	1.72985	0.214333	0.024223	0.171454	0.829542	1.62157
1.47856	1.5329	1.49454	0.0924477	1.73069	0.125487	0.128853	0.143763	0.736934	1.60313
1.47527	1.53468	1.49409	0.127746	1.72989	0.0794328	0.0998802	0.114888	0.776347	1.60306
1.46396	1.53862	1.48974	0.118144	1.7314	0.0964513	0.102915	0.0691874	0.874056	1.60569
1.45481	1.54393	1.48693	0.118707	1.7324	0.101981	0.101495	0.0949455	0.935804	1.60575
1.43346	1.55019	1.48068	0.120776	1.73111	0.096586	0.0916374	0.0953184	0.933715	1.59466
1.20971	1.64677	1.43158	0.101822	1.7345	0.0618675	0.0930848	0.10232	0.968584	1.48935
1.04326	1.71928	1.39709	0.082376	1.738	0.054666	0.0977362	0.105147	0.966427	1.39961
1.13908	1.68246	1.41883	0.0921987	1.73851	0.0722888	0.100901	0.101768	0.948655	1.43719
1.12106	1.69028	1.41491	0.0915587	1.73931	0.0651892	0.101779	0.0999675	0.944639	1.41778
1.0913	1.70285	1.4088	0.0928079	1.73907	0.0638715	0.0998951	0.100192	0.950897	1.38513
1.0816	1.70402	1.40493	0.102787	1.73593	0.0733594	0.104155	0.0991941	0.948222	1.26509
1.06228	1.70548	1.39643	0.123764	1.7291	0.0895759	0.109415	0.097633	0.954814	1.01738
1.09884	1.68921	1.40459	0.120453	1.72942	0.085724	0.105709	0.0980925	0.962737	1.09714
1.1287	1.67141	1.40962	0.125973	1.72745	0.0895339	0.102758	0.0985283	0.978727	1.07215
1.13935	1.65897	1.40926	0.132953	1.72396	0.0946116	0.100233	0.0995488	1.00048	0.993064
1.1313	1.66313	1.40808	0.129946	1.72503	0.0928254	0.100136	0.0997808	0.998759	1.00926
1.11851	1.66584	1.4061	0.125248	1.72554	0.0887418	0.100321	0.100416	0.99953	1.01049
1.12207	1.66384	1.40676	0.125916	1.72523	0.0893666	0.100156	0.100109	0.999734	1.00601
1.12277	1.65952	1.40716	0.123805	1.72471	0.0878334	0.0998837	0.0996539	1.00118	0.992234
1.12304	1.6582	1.40753	0.122013	1.72466	0.0866253	0.0998905	0.0997109	1.00048	0.996068
1.12264	1.65307	1.4087	0.116731	1.7243	0.0831372	0.0999728	0.0998285	1.0005	1.00024
1.12456	1.6462	1.41066	0.11098	1.72392	0.0794373	0.100064	0.0998522	0.999702	1.00894
1.12394	1.64145	1.41157	0.106938	1.72358	0.0765455	0.100009	0.09979	0.998829	1.00858
1.12352	1.63694	1.4121	0.104627	1.72313	0.0748228	0.0998115	0.0996826	0.999102	1.00201
1.12243	1.63283	1.41253	0.10186	1.72276	0.0728693	0.0997351	0.0996685	0.999346	0.999394
1.12032	1.63199	1.41224	0.10172	1.72272	0.0727684	0.0998227	0.0997439	0.999974	0.998619
1.11452	1.63264	1.41134	0.103792	1.72287	0.0743415	0.100225	0.100099	1.00138	1.00352
1.11622	1.62934	1.41215	0.102845	1.72259	0.0736944	0.100128	0.0999596	1.00082	1.00217
1.11537	1.61562	1.41449	0.0991387	1.72124	0.0710312	0.100007	0.0996647	0.999723	0.998847
1.09047	1.49983	1.43034	0.0803342	1.71122	0.0587846	0.0995922	0.0994806	0.999782	0.992456
1.08017	1.49004	1.43016	0.084844	1.71038	0.0616763	0.0999194	0.0996602	0.999951	0.994916
1.01499	1.37011	1.43916	0.101368	1.7	0.0727457	0.100714	0.100347	1.0003	1.00339
0.961418	1.23206	1.4531	0.10396	1.6878	0.0745284	0.101044	0.100555	1.0006	1.00446
0.956066	1.1837	1.45942	0.103403	1.68307	0.0744215	0.100698	0.100305	1.00113	1.00117
0.928947	1.06152	1.4738	0.103802	1.67149	0.0752396	0.100221	0.100022	1.00176	0.996309
0.932769	1.07624	1.47155	0.109744	1.6726	0.0796512	0.100105	0.0999784	1.00122	0.997609
0.945248	1.12122	1.4655	0.119206	1.67628	0.0865982	0.0999943	0.0999518	1.00022	1.00085
0.940352	1.09598	1.46865	0.11786	1.6739	0.0855609	0.100109	0.0999553	1.00037	1.00076
0.943059	1.10071	1.46803	0.11993	1.67416	0.0870626	0.100099	0.0999845	1.00004	1.00149
0.943834	1.09931	1.46805	0.121407	1.67385	0.0881719	0.100052	0.0999219	0.999864	1.00265
0.943671	1.09854	1.46813	0.121439	1.67377	0.0882019	0.100047	0.0999129	0.999854	1.00281
0.943541	1.09852	1.46812	0.121705	1.67376	0.0883905	0.100049	0.0998928	0.999731	1.00339
0.94333	1.09866	1.46808	0.121924	1.67378	0.0885383	0.100053	0.099879	0.99961	1.00394
0.943286	1.09876	1.46806	0.121952	1.67379	0.0885557	0.100053	0.0998756	0.999581	1.00407
0.943377	1.10134	1.46825	0.120505	1.67377	0.087548	0.100065	0.100007	0.999108	1.0047

Table 11: Steps taken using the method CD-UKF$_{\text{2nd numerical}}$ (72 steps)

k_{cat1}	K_{m1}	K_{m2}	V_4	K_4	k_4	k_5	k_6	V_1	K_{i1}
1.51656	1.54338	1.49568	0.0439102	1.72676	0.244499	1.00001×10^{-12}	0.12681	1.01535	1.67275
1.5159	1.54355	1.49548	0.130933	1.72579	0.175167	0.115212	0.117574	0.841063	1.64964
1.51358	1.54442	1.49489	0.160436	1.72407	0.139452	0.095189	0.10691	0.849064	1.64768
1.49215	1.55293	1.48928	0.156917	1.7235	0.126293	0.0986499	0.0649318	0.936681	1.64506
1.47223	1.56074	1.48408	0.155039	1.72373	0.120667	0.0975922	0.115501	0.97127	1.6387
1.46228	1.56449	1.48151	0.151046	1.72473	0.128347	0.0933146	0.100344	0.929983	1.63067
1.43594	1.5745	1.4747	0.146979	1.72477	0.118996	0.0946447	0.0918329	0.927093	1.61779
1.32051	1.61794	1.44496	0.126394	1.72595	0.0918702	0.0976024	0.0804568	0.929993	1.56143
1.14388	1.6835	1.39967	0.0950679	1.72787	0.0604789	0.100979	0.0811437	0.938808	1.47107
1.09754	1.70034	1.38783	0.0869091	1.72805	0.0631612	0.102158	0.0914973	0.947534	1.43901
1.08802	1.70361	1.38534	0.086463	1.72816	0.0603579	0.100758	0.102482	0.946115	1.41863
1.09551	1.70048	1.38721	0.0878092	1.72807	0.0628935	0.100767	0.10484	0.94943	1.40619
1.10815	1.69326	1.39023	0.0910273	1.72801	0.0687464	0.100684	0.112488	0.965595	1.30726
1.13017	1.67857	1.3954	0.0980892	1.72801	0.0781234	0.100587	0.120812	0.998216	1.05812
1.14433	1.672	1.39903	0.101538	1.72793	0.0783941	0.100253	0.113024	1.00288	1.02884
1.14258	1.67093	1.39878	0.103137	1.72784	0.0739765	0.0996208	0.0975603	1.00082	1.01336
1.13611	1.67233	1.39717	0.102656	1.72783	0.0735534	0.0999488	0.0999879	1.00172	0.988893
1.13518	1.67268	1.39703	0.102489	1.72779	0.0732453	0.100002	0.100066	0.999447	1.00478
1.13537	1.67204	1.39717	0.102877	1.72774	0.0735194	0.0999407	0.0997354	0.99963	1.00337
1.13501	1.67052	1.39735	0.103767	1.72759	0.0741438	0.0999024	0.0994936	0.999755	1.00185
1.1328	1.66429	1.39798	0.107338	1.72696	0.0766406	0.0998404	0.0990234	0.99996	0.998983
1.12809	1.65296	1.39902	0.113578	1.72578	0.0809933	0.099832	0.0987287	1.00004	0.997342
1.12039	1.63591	1.40049	0.122417	1.72398	0.0871539	0.0999229	0.0988787	0.999891	0.998519
1.11433	1.62313	1.40154	0.128148	1.72263	0.0911555	0.100065	0.099444	0.999579	1.0021
1.11138	1.61686	1.40207	0.129785	1.722	0.0923108	0.100164	0.0999368	0.999341	1.00509
1.10851	1.61005	1.40269	0.130077	1.72137	0.0925354	0.100256	0.100427	0.999115	1.00807
1.10217	1.59326	1.40435	0.129639	1.71988	0.0922805	0.10039	0.101174	0.998778	1.01258
1.0866	1.54906	1.4089	0.128393	1.71601	0.091544	0.100581	0.102268	0.998272	1.01932
1.04889	1.43757	1.42062	0.126134	1.7063	0.090303	0.100825	0.103742	0.997556	1.02868
0.980928	1.22982	1.44282	0.123015	1.68828	0.0887458	0.100979	0.104826	0.996963	1.03607
0.941164	1.09951	1.45718	0.119681	1.6771	0.0868	0.100748	0.103734	0.997498	1.0292
0.939154	1.08221	1.45953	0.116021	1.6758	0.084265	0.100152	0.100554	0.999261	1.00768
0.944237	1.09936	1.45757	0.118562	1.67717	0.0859566	0.100045	0.0999217	0.999562	1.00381
0.943379	1.09562	1.45802	0.119661	1.67677	0.0867699	0.100046	0.0999097	0.999539	1.00394
0.943149	1.09525	1.45802	0.120008	1.67667	0.0870402	0.10005	0.0999261	0.999528	1.00406
0.943246	1.0954	1.45798	0.120075	1.67661	0.0870955	0.100051	0.0999303	0.999525	1.00408
0.943579	1.09591	1.45767	0.120381	1.67599	0.087362	0.100056	0.0999484	0.999513	1.00418
0.944038	1.09641	1.45689	0.120786	1.67421	0.0877516	0.100064	0.0999744	0.999494	1.00434
0.945057	1.09741	1.45445	0.121439	1.66856	0.0884851	0.100077	0.100021	0.999463	1.00459
0.946722	1.098	1.44796	0.122354	1.65305	0.0897996	0.1001	0.100095	0.999414	1.00498
0.950328	1.09778	1.42912	0.123602	1.60768	0.0925207	0.100139	0.100224	0.999337	1.00561
0.960175	1.0926	1.36438	0.125141	1.45085	0.0998923	0.100224	0.1005	0.999202	1.00676
0.967558	1.08914	1.31563	0.123789	1.33307	0.103872	0.100271	0.10065	0.99917	1.00711
0.975677	1.08799	1.2679	0.119428	1.21858	0.105801	0.100308	0.100771	0.999196	1.00707
0.991704	1.08595	1.17778	0.106687	1.00243	0.106461	0.100349	0.100887	0.999375	1.00605
0.988939	1.06181	1.15742	0.100486	0.945359	0.104767	0.100216	0.100387	0.999751	1.00242
0.977775	1.02716	1.14959	0.102545	0.915576	0.107507	0.100027	0.0996889	1.00006	0.998549
0.973065	1.04319	1.18877	0.106278	1.01578	0.105606	0.100071	0.0998576	0.999852	1.00055
0.977508	1.04614	1.16858	0.103015	0.968617	0.105244	0.100135	0.100077	0.999807	1.0015
0.975751	1.03975	1.1674	0.102549	0.963682	0.104998	0.100122	0.100031	0.999847	1.00111
0.973874	1.0352	1.16785	0.101674	0.963508	0.103968	0.100099	0.099961	0.999896	1.00065
0.973238	1.03537	1.1704	0.100161	0.970002	0.101936	0.10009	0.0999336	0.99992	1.00052
0.974623	1.04029	1.17161	0.0990035	0.974744	0.10049	0.100094	0.0999473	0.999913	1.00069
0.975561	1.04368	1.173	0.0988976	0.979223	0.100159	0.100099	0.0999653	0.999902	1.00084
0.975749	1.0444	1.17335	0.0989757	0.980317	0.100195	0.100101	0.0999696	0.999897	1.00089
0.975696	1.04467	1.17401	0.0990676	0.982053	0.10019	0.100101	0.0999706	0.999895	1.0009
0.975598	1.04439	1.174	0.0990683	0.981976	0.100195	0.100101	0.0999704	0.999896	1.0009
0.975514	1.04401	1.17371	0.0990366	0.981209	0.100207	0.1001	0.0999697	0.999897	1.00088
0.975931	1.04532	1.1733	0.098953	0.98258	0.100034	0.100098	0.0999651	0.999841	1.00125
0.975977	1.04519	1.17309	0.0989875	0.982664	0.100068	0.1001	0.099969	0.999859	1.00112
0.976251	1.04303	1.16929	0.0990991	0.980549	0.100325	0.100109	0.0999879	0.999949	1.00047
0.977284	1.04274	1.16415	0.0992097	0.98194	0.100364	0.100113	0.0999955	0.999989	1.00018
0.984335	1.0386	1.12468	0.0998183	0.992349	0.100407	0.100128	0.100029	1.00015	0.999057
0.98905	1.03169	1.09204	0.100041	0.99752	0.100331	0.100134	0.100038	1.00016	0.999047
0.995026	1.01813	1.04424	0.100219	1.00191	0.100236	0.100138	0.100038	1.0001	0.999553
0.997734	1.00375	1.013	0.100072	1.00091	0.100097	0.100131	0.100014	0.999918	1.00093
0.997429	0.999672	1.00902	0.100047	0.997898	0.100233	0.100128	0.0999976	0.99987	1.00116
0.997578	0.999219	1.00741	0.100028	0.997143	0.100257	0.100129	0.0999986	0.99987	1.00112
0.997152	1.0001	1.01053	0.0999663	0.995677	0.100279	0.10013	0.100002	0.999873	1.00108
0.997344	0.999576	1.00892	0.0999695	0.995677	0.100282	0.100129	0.1	0.999871	1.00111
0.997462	0.999548	1.00838	0.0999778	0.995994	0.100272	0.100129	0.0999995	0.99987	1.00112
0.99737	0.999836	1.00919	0.0999765	0.99599	0.100271	0.100129	0.0999992	0.999871	1.00112

B Differentiating the CD-EKF with respect to the parameters θ

In Section 4 the differentiated CD-UKF was derived and it was explained how it was used in the minimization of the objective function $V(\theta;\mathscr{Y}_N)$, given in (3.1). In the benchmarking in Section 5, it was further stated that one of the methods used was the CD-EKF with analytical gradient. If one switches from using the CD-UKF to the CD-EKF in the parameter estimation framework, one must also switch from using the differentiated CD-UKF to the differentiated CD-EKF. We state here the symbolic differentiation of the CD-EKF given in Section 2.4.3.

- *Prediction step*

  The derivative of (2.19) with respect to θ_l is

$$\frac{d}{d\theta_l}\left(\frac{dm_t}{dt}\right) = \frac{d}{dt}\left(\frac{dm_t}{d\theta_l}\right) = \frac{\partial f(m_t,\theta)}{\partial m_t}\frac{dm_t}{d\theta_l} + \frac{\partial f(m_t,\theta)}{\partial \theta_l}. \tag{B.1}$$

  The derivative of (2.20) with respect to θ_l is

$$\frac{d}{d\theta_l}\left(\frac{dP_t}{dt}\right) = \frac{d}{dt}\left(\frac{dP_t}{d\theta_l}\right) = \frac{dF_x(m_t,\theta)}{d\theta_l}P_t + F_x(m_t,\theta)\frac{dP_t}{d\theta_l} + \frac{dP_t}{d\theta_l}F_x(m_t,\theta)^T + \tag{B.2}$$
$$P_t\frac{dF_x(m_t,\theta)}{d\theta_l}^T + \frac{\partial L(\theta)}{\partial\theta_l}L(\theta)^T + L(\theta)\frac{\partial L(\theta)^T}{\partial\theta_l},$$

  where
$$\frac{dF_x(m_t,\theta)}{d\theta_l} = \frac{\partial F_x(m_t,\theta)}{\partial m_t}\frac{dm_t}{d\theta_l} + \frac{\partial F_x(m_t,\theta)}{\partial\theta_l}.$$

- *Update step*

  The derivatives of (2.21) and (2.22) are

$$\frac{d\mu_k}{d\theta_l} = \frac{\partial h(m_k^-,\theta)}{\partial m_k^-}\frac{dm_k^-}{d\theta_l} + \frac{\partial h(m_k^-,\theta)}{\partial\theta_l} \tag{B.3}$$

  and

$$\frac{dS_k}{d\theta_l} = \frac{dH_x(m_k^-,\theta)}{d\theta_l}P_k^- H_x(m_k^-,\theta)^T + H_x(m_k^-,\theta)\frac{dP_k^-}{d\theta_l}H_x(m_k^-,\theta)^T + \tag{B.4}$$
$$H_x(m_k^-,\theta)\frac{dH_x(m_k^-,\theta)}{d\theta_l}^T + \frac{dR_k(\theta)}{d\theta_l},$$

  respectively, where

$$\frac{dH_x(m_k^-,\theta)}{d\theta_l} = \frac{\partial H_x(m_k^-,\theta)}{\partial m_k^-}\frac{dm_k^-}{d\theta_l} + \frac{\partial H_x(m_k^-,\theta)}{\partial\theta_l}.$$

Furthermore, the derivatives of (2.23)-(2.25) are

$$\frac{dK_k}{d\theta_l} = \frac{dP_k^-}{d\theta_l} H_x(m_k^-, \theta)^T S_k^{-1} + P_k^- \frac{dH_x(m_k^-, \theta)^T}{d\theta_l} S_k^{-1} + \tag{B.5}$$

$$P_k^- H_x(m_k^-, \theta)^T (-S_k^{-1} \frac{dS_k}{d\theta_l} S_k^{-1}),$$

$$\frac{dm_k}{d\theta_l} = \frac{dm_k^-}{d\theta_l} + \frac{dK_k}{d\theta_l}(y_k - \mu_k) - K_k \frac{d\mu_k}{d\theta_l} \tag{B.6}$$

and

$$\frac{dP_k}{d\theta_l} = \frac{dP_k^-}{d\theta_l} - \frac{dK_k}{d\theta_l} S_k K_k^T - K_k \frac{dS_k}{d\theta_l} K_k^T - K_k S_k \frac{dK_k^T}{d\theta_l}, \tag{B.7}$$

respectively.

References

[1] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial, 2005.

[2] Michael T. Heath. *Scientific Computing - An Introductory Survey.* McGraw-Hill, 2002.

[3] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, April 1970.

[4] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, pages 401–422, 2004.

[5] Niels R. Kristensen and Henrik Madsen. *Continuous Time Stochastic Modelling, CTSM 2.3 - Mathematics Guide.*

[6] Niels R. Kristensen, Henrik Madsen, and Sten B. Jørgensen. A method for systematic improvement of stochastic grey-box models. *Computers and Chemical Engineering*, 28(8):1431 – 1449, 2004.

[7] Niels R. Kristensen, Henrik Madsen, and Sten B. Jørgensen. Parameter estimation in stochastic grey-box models. *Automatica*, 40(2):225–237, February 2004.

[8] Lennart Ljung and Torkel Glad. *Modeling of Dynamic Systems (Prentice Hall Information and System Sciences Series).* Prentice Hall PTR, April 1994.

[9] Carmen G. Moles, Pedro Mendes, and Julio R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome research*, 13(11):2467–2474, November 2003.

[10] Kaare B. Petersen and Michael S. Pedersen. The matrix cookbook, oct 2008.

[11] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 2nd edition, October 1992.

[12] John A. Rice. *Mathematical Statistics and Data Analysis.* Thomson Brooks/Cole, 2007.

[13] Maria Rodriguez-Fernandez, Pedro Mendes, and Julio R. Banga. A hybrid approach for efficient and robust parameter estimation in biochemical pathways. *Biosystems*, 83(2-3):248 – 265, 2006. 5th International Conference on Systems Biology - ICSB 2004.

[14] Simo Särkka. *Recursive Bayesian Inference On Stochastic Differential Equations.* PhD thesis, Helsinki University of Technology, 2006.

[15] Simo Särkka. On unscented Kalman filtering for state estimation of Continuous-Time nonlinear systems. *Automatic Control, IEEE Transactions on*, 52(9):1631–1641, 2007.

[16] Niklas Skaar. Parameter estimation methods for continuous time dynamical systems given discrete time measrurments. Master's thesis, Chalmers university of technology, Gothenburg, Sweden, 2008.

[17] Zhen Sun and Zhenyu Yang. Study of nonlinear parameter identification using UKF and maximum likelihood method. In *Control Applications (CCA), 2010 IEEE International Conference on*, pages 671 –676, September 2010.

[18] Rudolph van Der Merwe and Eric A. Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *in International Conference on Acoustics, Speech, and Signal Processing*, pages 3461–3464, 2001.

[19] Eric A. Wan and Rudolph van der Merwe. The unscented Kalman filter for nonlinear estimation. pages 153–158, August 2002.

[20] Inc Wolfram Research. Wolfram Mathematica 8 documentation center, 2011.